
# UNIVERSIDAD DE MURCIA

## FACULTAD DE INFORMÁTICA

Proyecto Final de Carrera

# SISTEMA VEHÍCULAR CONTEXT-AWARE BASADO EN PROCESAMIENTO DE EVENTOS COMPLEJOS PARA LA DETECCIÓN DE ITINERARIOS A PARTIR DE INTERACCIONES USUARIO/CARROCERÍA. PROPUESTA DE UN ALGORITMO PARA EL MODELADO DE USUARIO

*Alumna*:
Lidia Fernández Carpes

*Directores*:
Mercedes Valdés Vela
Fernando Terroso Sáenz

Julio 2010

# Índice de Contenido

# Índice de Figuras

# Índice de Tablas

# Resumen

Los vehículos de hoy en día tienen un elevado número de sensores capaces de proporcionar gran cantidad de información de diversa naturaleza. En particular, los llamados *sensores de carrocería*, perciben las interacciones de los pasajeros con elementos tales como puertas, ventanillas, cinturones de seguridad, etc. En este trabajo, proponemos la idea de que con el adecuado procesamiento de los datos procedentes de tales sensores se puede inferir información útil sobre los hábitos que un determinado usuario tiene al utilizar su vehículo. En particular, en este trabajo proponemos un sistema basado en la tecnología de *Procesamiento de Eventos Complejos* (CEP, Complex Event Processing) que, a partir de las interacciones de los pasajeros con los elementos mencionados, es capaz de detectar la ocupación de un vehículo así como los itinerarios que suele seguir. En este sentido se trata de un sistema capaz de percibir el contexto (*context-aware)* interno del vehículo.

El sistema incorpora mecanismos para que el usuario corrija las inferencias sobre la ocupación del vehículo detectada automáticamente mediante CEP. Estas correcciones son la entrada a un algoritmo que proponemos como una de las aportaciones fundamentales del presente trabajo, cuyo objetivo es la adaptación on-line del perfil del usuario a partir de la experiencia. En este sentido se trata de un *sistema adaptativo.* Además, la aplicación proporciona las funcionalidades necesarias para etiquetar y almacenar datos sobre interacciones de cada usuario con su vehículo con el propósito de realizar tareas de aprendizaje supervisado off-line una vez se tengan suficientes datos.

El presente trabajo constituye una pequeña contribución en la investigación de los Sistemas Inteligentes de Transporte (ITSs , Intelligent Transport Systems) en general y en la mejora de la Percepción de la Situación en ITSs en particular.

# 1 Ámbito de Investigación

Los objetivos y la motivación del presente trabajo no se pueden entender bien sin antes describir brevemente la línea de investigación en el que se ha desarrollado el proyecto. Así que, en esta sección se describe brevemente esta línea.

Este proyecto se ha llevado a cabo como parte de un trabajo en el que se pretende desarrollar un sistema *context-aware (consciente del contexto)* [1].En general, los sistemas context-aware son capaces de:

- por un lado, percibir su entorno y,
- por otro, adaptarse a las circunstancias percibidas, de modo que proporcionen funcionalidades de forma dinámica adecuándose a cada situación. Esta característica de adaptación también se suele referir a la capacidad de algunos sistemas de modelar, en base al aprendizaje a partir de la experiencia que se adquiere en la interacción con el usuario, cómo adecuarse mejor a sus necesidades. Es por ello por lo que también se les suele denominar *sistemas adaptativos*.

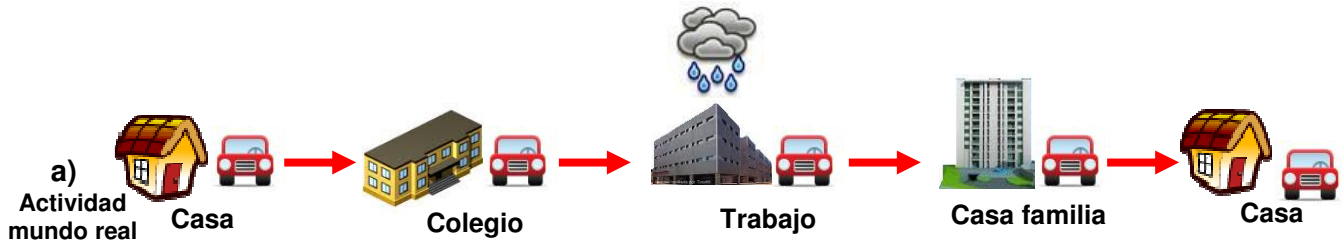
Fundamentalmente, hay cuatro dimensiones o categorías en las que se puede dividir el contexto a percibir por un sistema de este tipo: *identidad, localización, actividad* y *tiempo*. Pues bien, el objetivo final de la **línea de investigación** en la que se enmarca este proyecto es el desarrollo de un sistema context-aware que iría instalado en el ordenador de a bordo de un vehículo. El sistema, a partir de las interacciones de los pasajeros con la carrocería del vehículo y otras fuentes de información externas (sistemas de información geográfica, mapas, información meteorológica, etc.), sería capaz de detectar las cuatro categorías contextuales anteriores y de hacer sugerencias al usuario en base a las circunstancias percibidas. En particular se persigue que el sistema sea capaz de:

- Detectar detalles de la ocupación del coche como, por ejemplo, número de pasajeros, rangos de edades, género, disposición en los asientos (percepción de la *identidad*).
- Identificar el origen y destino espacial del itinerario y contextualizarlo, ¿estamos en casa?, ¿en un parking?, etc. (percepción de la *localización*)
- Identificar el momento inicial y final (hora y fecha) del itinerario y contextualizarlo (percepción del *tiempo*)

- Percibir la actividad como, por ejemplo, si estamos yendo al trabajo, recogiendo a los niños, yendo de compras... (percepción de la *actividad*)
- Adaptarse a las circunstancias para proporcionar al usuario información o sugerencias. Así, si se percibe que se está desarrollando la actividad de *llevar los niños al colegio*, podría sugerir caminos alternativos cuando se detectan situaciones de atasco etc.
- Aprender las rutinas de un usuario, en cuanto para qué y en qué circunstancias usa el coche. Por ejemplo, el sistema sería capaz de percibir rutinas tales como que *el usuario X sale de su domicilio todos los lunes sobre las 18h acompañado de dos niños y regresa sobre las 19:30.*

La Figura 1 a) muestra el itinerario real que sigue un usuario con su coche a lo largo de un día. Derivan de este itinerario dos líneas distintas de detalle en función de la información del itinerario mostrado que se es capaz de percibir de las categorías en las que se ha visto que se divide la información contextual.

- I = identidad
- L = localidad
- T = tiempo
- A = actividad




| | | | | |
|---|---|---|---|---|
| **b) Identificación** | I: Adulto, niño<br>L: 345,654<br>T: 09:02 | I: Adulto<br>L: 675,897<br>T: 09:34 | I: Adulto<br>L: 564,-786<br>T: 15:12 | I: 2Adultos, niño<br>L: 453,234<br>T: 18:45 |
| **c) Contextualización** | I: Padre, Hijo menor<br>L: Casa<br>T: Temprano<br>A: Ir al colegio | I: Padre<br>L: Colegio<br>T: Mañana<br>A: Ir al trabajo | I: Padre<br>L: Trabajo<br>T: Mediodía lluvioso<br>A: Ir a comer | I: 2Adultos, niño<br>L: 453,234<br>T: Por la tarde<br>A: Regresar a casa |

Figura 1. Categorías de percepción

Para lograr los objetivos descritos se necesita resolver una serie de cuestiones de diferentes niveles de abstracción. Las abordadas en este proyecto son las más cercanas al vehículo mostradas en la Figura 1 b) (identificar ocupación, identificar puntos espaciales y temporales de origen y destino). Las cuestiones de alto nivel concernientes a la contextualización de la información identificada mostradas en la Figura 1 c), requiere del uso de fuentes de información externa y de dotar al sistema de capacidades de razonamiento semántica que queda fuera del presente trabajo.

# 2 Motivación

Como se acaba de explicar, uno de los motivos principales para la realización de este proyecto es avanzar un paso en el trabajo de investigación descrito anteriormente. No obstante, por sí sólo también tiene interés como se verá a continuación.

Hoy día, en las centrales de tráfico existen sistemas de información que monitorizan la densidad del tráfico, maniobras e infracciones del los vehículos, accidentes y atascos entre otras cosas. Se puede suponer que, en estos sistemas, cada vehículo circulando por una autovía es representado en la pantalla como algo así como un *punto sobre un mapa*, como se aprecia en la Figura 2.


Figura 2. Representación de vehículo sobre mapa

Dejando a parte la controversia de la privacidad, se cree que sería interesante que tales sistemas tuvieran un poco más de información sobre cada vehículo. En particular, sería interesante conocer algo del pasaje (número de ocupantes, cuales de ellos son adultos y cuales niños,…) y mostrar ese punto en el mapa como en la Figura 3.


Figura 3. Representación de vehículo con ocupación sobre mapa

Aunque los objetivos particulares de este trabajo se describirán en una sección posterior, se puede tomar esta sencilla idea como la meta fundamental.

A continuación se exponen algunos ejemplos que ponen de manifiesto el interés de la idea expuesta.

- Labores de Rescate: Nada más que los cinco primeros meses de 2010, ha habido en España 550 accidentes mortales [2]. Ante un accidente con varios vehículos implicados, se cree que si se supiera de antemano el número, posición y tipo de pasajeros en cada uno de los vehículos implicados, quizás se podrían aplicar mejor los protocolos de rescate (número de ambulancias a avisar, herramientas para rescatar a personas atrapadas, etc.).

- Olvido de Niños e Imprudencias: El hecho de que desde una central de tráfico se tenga información ocupacional de un vehículo, permitiría detectar situaciones en las que menores son olvidados dentro del mismo. Aunque infrecuentes, estos despistes pueden desencadenar consecuencias trágicas y ocurren más a menudo de lo debido. Se pueden encontrar muchos sucesos acerca de este tipo de hechos. Basta con introducir en www.google.es las palabras clave: niño, olvidado, coche, y salen 131.000 resultados.

Aunque ya se han propuesto diversos sensores que alertan de este tipo de olvidos como [3] y [4], tales dispositivos sólo sirven cuando se trata de un olvido, pero no en los casos en los que exista una imprudencia deliberada (se sale del coche pensando en volver en 2 minutos y algo impide que se vuelva).

# 3 Trabajos Relacionados

Con el fin de situar al lector, se considera interesante describir brevemente algunos trabajos relacionados y enmarcar el presente proyecto entre ellos.

Este trabajo se encuadra en el área de los Sistemas Inteligentes de Transporte (ITS, Intelligent Transport System) [5]. Los ITS constituyen una variedad de herramientas provenientes de la Ingeniería de Tráfico y de las Tecnologías de la Información y de la Comunicación que actúan unidas con el propósito de ofrecer unos sistemas de transporte más eficientes y seguros [6]. Se espera que el desarrollo y la aplicación de los ITS sirvan para reducir el número y gravedad de los accidentes automovilísticos y para aumentar la seguridad en el transporte rodado.

Una de las líneas de trabajo que existe en ITS es la destinada a la mejora de la denominada *Percepción de la Situación (SA, Situation Awareness)*. Este es el nombre más comúnmente usado en el área de ITS para el concepto de Percepción de Contexto descrito en la sección 1. El objetivo de la SA es que los elementos presentes en la carretera sean capaces de reconstruir la escena en la que se encuentran y puedan así advertir posibles peligros que existen en el entorno. En este sentido, el presente proyecto y la línea de investigación descrita en la sección 1 forman parte de este tipo de líneas de trabajo que persiguen la mejora de la SA.

En la Tabla 1 se muestran algunos trabajos en la línea de SA, clasificados teniéndose en cuenta:

- El tipo de SA que generan: Interna, es decir referida al entorno restringido al propio vehículo, o externa al coche, representando a todos los elementos externos que rodean al vehículo (carretera, vehículos cercanos, climatología, etc.).
- El tipo de información de entrada necesaria para generar la SA: Sistemas de información internos al coche (datos de los sensores del vehículo, de sus ocupantes, etc.) o externos (mensajes de otros vehículos, servicios web, centrales de tráfico, etc.).
- Dónde se visualizará la información generada: En un sistema a bordo o en un sistema externo al coche, como por ejemplo, un sistema de control de tráfico de la Dirección General de Tráfico (DGT).
- Tecnologías implicadas para desarrollar la solución.

| | SA<br>Interna/externa | Información de Entrada<br>Interna/Externa | Donde<br>Central/A bordo | Tecnologías Implicadas |
|---|---|---|---|---|
| **[7]** | Interna | Interna | A bordo | Neuro-Fuzzy |
| **[8]** | Interna | Interna | A bordo | Redes Neuronales |
| **[9]** | Externa | Interna / Externa | A bordo | Arquitectura |
| **[10]** | Externa | Externa | Central | CEP |
| **[11]** | Externa | Interna / Externa (VANET) | A bordo | CEP |
| **Línea de Investigación de Sección 1** | Interna / Externa | Interna / Externa | A bordo | CEP, Aprendizaje, Semántica |
| **PFC** | **Interna** | **Interna** | **A bordo** | **CEP, Mecanismo de adaptación** |

Tabla 1. Trabajos enmarcados en la línea de trabajo de mejora del SA

Así, en [7] se propone un Sistema de Clasificación Basado en Reglas Difusas (FRBCS, Fuzzy Rule Based Clasification System) que, tomando como entrada los datos de diferentes sensores del vehículo tales como la odometría, y a partir de un conjunto de reglas difusas, es capaz de inferir el tipo de maniobra que se lleva a cabo en cada momento (parado, aceleración, deceleración o en velocidad de crucero). En [8] se persigue el mismo objetivo que en [7] aunque utilizando redes neuronales para su consecución. Como se puede ver, ambos enfoques utilizan como información de entrada datos provenientes únicamente del interior del vehículo como son los sensores del mismo y generan una información restringida al vehículo, en concreto la maniobra que el conductor está realizando en cada momento.

En [9] se define una plataforma integral para la provisión de servicios telemáticos tradicionales y de carácter ubicuo en un entorno ITS. Dicha plataforma propone un mecanismo de a bordo enfocado a recibir información contextual de diferentes fuentes externas y, a partir de ellas, proporcionar sugerencias a los usuarios del vehículo

teniendo en cuenta sus preferencias, las cuales han sido introducidas previamente en el sistema. Dichas sugerencias pueden abarcar un gran número de aspectos, tales como tipos de hoteles en los que el usuario puede estar interesado en alojarse, restaurantes que se ajustan al presupuesto económico del usuario, etc. Por tanto, en este trabajo se recoge información de fuentes externas (tales como servicios web) e internas (las preferencias del usuario) y genera una información del contexto externo al vehículo, concretamente, lugares en los que el usuario puede estar interesado en visitar.

[10] propone una arquitectura guiada por eventos para la gestión del tráfico por carretera. Dicha arquitectura recibe información de unos sensores repartidos por la carretera, los cuales miden la intensidad del flujo de tráfico. A partir de esta información se generan una serie datos que son tomados como eventos por el sistema. Estos eventos se procesan para detectar problemas de tráfico y proponer posibles soluciones a los mismos. Como se puede ver, la fuente de información de entrada en este enfoque es externa al vehículo (datos de los sensores de la carretera) así como la salida que genera, ya que informa del estado del tráfico en una carretera.

La propuesta presentada en [11] se centra en la detección de situaciones peligrosas para los ocupantes del vehículo como son posibles colisiones con otros vehículos o atascos mediante el procesamiento de los mensajes de una red VANET (Vehicular Ad-Hoc Network), los cuales contienen información acerca de la posición y velocidad de su vehículo emisor. En este caso se propone una aplicación de a bordo que recibe tanto los mensajes del vehículo en donde se está ejecutando como de los vehículos cercanos a este. Todos estos mensajes son procesados como eventos, a partir de los cuales se detectan situaciones peligrosas anteriormente comentadas. Por lo tanto, se puede apreciar que este sistema utiliza información de entrada tanto interna (mensajes del vehículo en el cual se ejecuta) como externa (mensajes de vehículos vecinos) generando información externa al vehículo ya que los riesgos de colisión y atascos se refieren a situaciones que afectan al vehículo pero ocurren fuera de el.

El presente trabajo se centra en el desarrollo de un sistema que se incorporará a bordo del vehículo. Este sistema recoge información de los datos de los sensores del automóvil. Dicha información será procesada para inferir itinerarios, tramos y la ocupación del vehículo en cada tramo. Por lo tanto, este es un sistema en el que se recoge información interna del coche (sus sensores) para genera como salida información también interna del mismo (la ocupación del habitáculo). Hasta ahora,

aunque diversos trabajos de los mencionados tengan en cuenta información proveniente del sistema de sensores del vehículo, esta no se había utilizado para realizar inferencias sobre el habitáculo del mismo.

El presente proyecto es el primer paso de una línea de trabajo más ambiciosa (ya descrita en la sección 1), donde se pretende contextualizar al usuario con su vehículo. El objetivo es tomar como entrada información tanto interna (sensores del automóvil explicados anteriormente) como de fuentes externas con el objetivo de contextualizar al usuario en el entorno. La salida generada será interna, al informar del habitáculo interior del vehículo con la salvedad de que en este caso se generará la información contextualizada, es decir, en vez situar a un adulto en el departamento del conductor de un vehículo, situará al "padre" o en vez de ver que hay un niño en el coche, verá al "hijo menor". Pero además, la información generada por el sistema también será externa al vehículo, informando al usuario sobre datos de interés que hay en el entorno fuera del coche, tales como posibles atascos en el itinerario que está realizando, o posibles zonas disponibles para aparcar.

# 4 Objetivos

Teniendo en cuenta todo lo expuesto, el objetivo del proyecto actual es el diseño e implementación de un prototipo para una aplicación de a bordo con las siguientes funcionalidades:

- Recolección de datos: el sistema debe ser capaz de recopilar la información sobre las interacciones usuario-carrocería detectadas en tiempo real por los sensores del coche.

- Etiquetado y supervisión: el sistema debe ofrecer mecanismos para que el usuario actúe como supervisor, y por tanto sea capaz de etiquetar, componer y agrupar las interacciones usuario-carrocería percibidas por los sensores, en unidades de información de nivel medio, llamadas aquí *ocupación*, *tramos* e *itinerarios*. Estos términos se verán a lo largo de toda la documentación, se puede profundizar en ellos en la sección 6.2 a), b) y c).

- Persistencia: el prototipo debe proporcionar un mecanismo para registrar en un sistema de almacenamiento, tanto los elementos de ocupación, tramo e itinerario construidos por el usuario como las interacciones usuario-carrocería que forman parte de los mismos. Con ello se pretende recoger datos útiles para realizar tareas de aprendizaje Computacional supervisado off-line.

- Procesamiento de Eventos: el sistema, sin supervisión por parte del usuario y, tomando únicamente como entrada la información de bajo nivel obtenida por las interacciones del usuario con la carrocería del vehículo, debe detectar tramos, itinerarios y ocupación por sí solo. Para ello, monitorizará las interacciones usuario-carrocería en busca de patrones que indiquen la ocupación que puede haber en el habitáculo, tramos o itinerarios. Dichos patrones son definidos en tiempo de diseño.

- Consolidación y/o Refutación de Patrones: el sistema debe ofrecer los métodos necesarios para evaluar la confiabilidad de los patrones prediseñados. Para ello, se sirve de la comparación de los tramos, itinerarios y ocupación inferidos automáticamente por el sistema con los proporcionados por el usuario.

- Adaptación y Modelado de Usuario: Esta funcionalidad está muy relacionada con la anterior. Es posible que los patrones predefinidos sean válidos para unos usuarios y no para otros. De hecho, es más que probable que la forma de operar

un mismo usuario sea distinta en según que circunstancias (sirva como ejemplo el del copiloto que se pone el cinturón sólo para trayectos por autovía). Por tanto, la aplicación debe adaptarse a un usuario concreto, penalizando aquellos patrones que se suelan equivocar y recompensando aquellos que sí funcionen. Además, este perfil de usuario debe poder cargarse y guardarse cada vez que el usuario entre y salga del sistema.

# 5 Tareas y Tecnologías Empleadas

Cada uno de las funcionalidades requeridas para el prototipo, detalladas en la sección anterior, implican la realización de una serie de tareas para su consecución. Estas tareas se describen en esta sección así como las tecnologías usadas para llevarlas a cabo.

Para una mayor claridad, en la Figura 4 se muestra la arquitectura conceptual del sistema. Cada módulo está asociado a alguna de las funcionalidades mencionadas.

El funcionamiento general es el siguiente: el sistema a desarrollar tomará en cuenta las interacciones que un usuario realiza con su vehículo (MRAC). Para este proyecto serán relevantes únicamente las interacciones de carrocería (abrir y cerrar puertas, ventanillas, cinturones,…), no teniéndose en cuenta las actividades que tengan que ver con la conducción (volante, marchas, temperatura del motor,…). Esto es debido a que es a partir de las actividades con las que los usuarios interaccionan con la carrocería del vehículo, por la que se pueden inferir datos sobre el pasaje de un coche. Si por ejemplo, se percibe que una puerta de un determinado departamento del coche se abre y después se cierra y además percibe que el cinturón de ese departamento se ha anclado, se va a suponer que se ha montado una persona. Estas actividades del mundo real son percibidas por los sensores dando lugar a eventos.

A partir de esos eventos se constituirán dos flujos de ejecución. Por un lado, el usuario agrupará los eventos formando con ellos tramos e itinerarios (MIIU) que serán almacenados para su posterior consulta (MSAITE). Por otro lado, los eventos sin etiquetar entrarán en un sistema que tratará de inferir esos mismos tramos e itinerario por sí sólo (MGIA). Las inferencias generadas se mostrarán al usuario para que las pueda validar (MMUC), y en función de sus validaciones, se irá creando un perfil por usuario, el cual representa sus hábitos y costumbres. Este perfil será almacenado en el sistema (MSAPU) para tenerlo en cuenta en futuras inferencias.

Las inferencias realizadas por el sistema se mostrarán al usuario el cual podrá corregirlas. A partir de las modificaciones que realice el usuario se irá modelando el perfil del mismo en el sistema, al tiempo que se refina el mecanismo de inferencia.

Vista la relación entre los diferentes módulos requeridos, se examinan las diferentes tareas a realizar, y en su caso las tecnologías, con el objetivo de cumplir cada módulo individualmente.

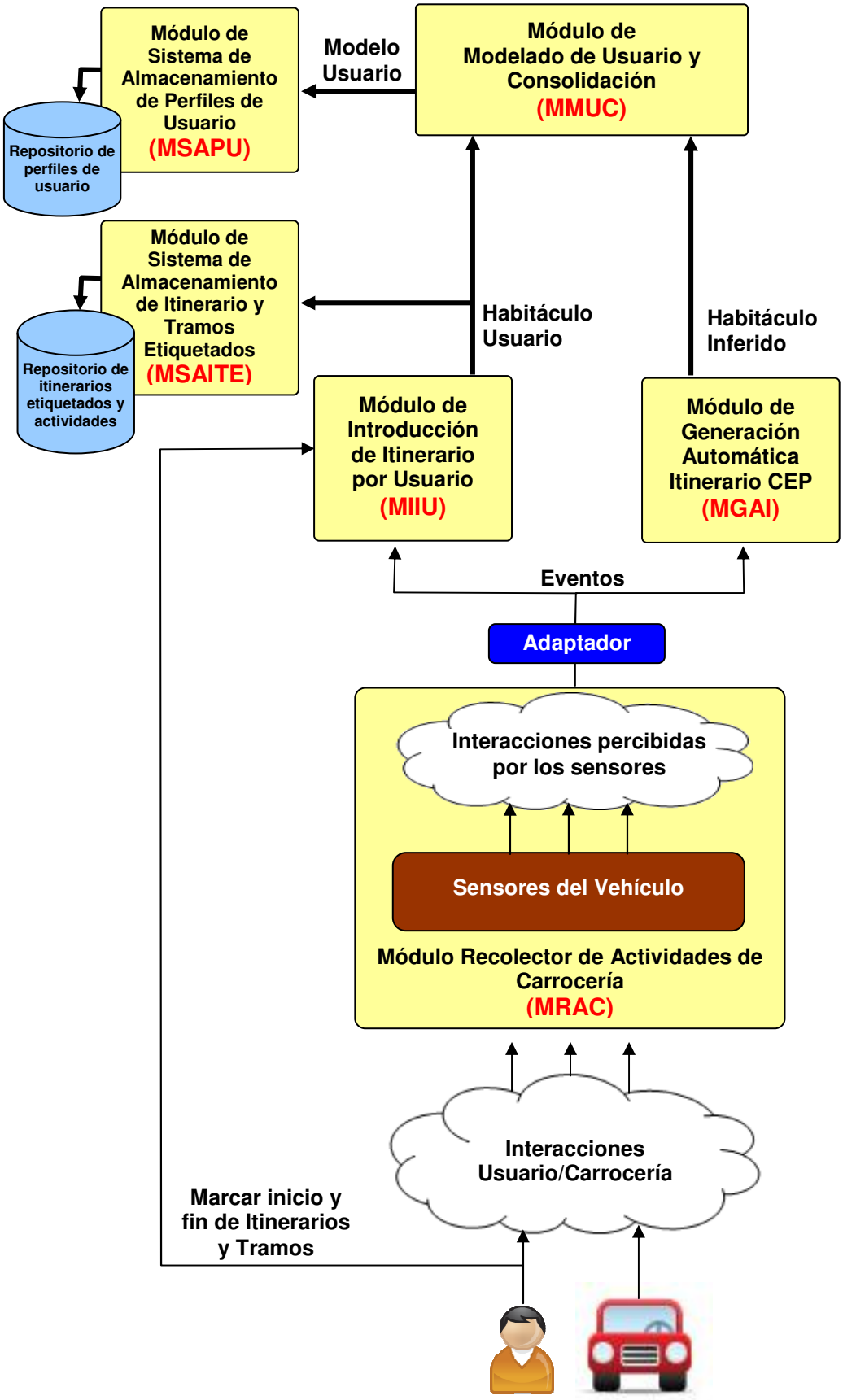



Figura 4. Estructura de los módulos de la aplicación

## 5.1 Módulo Recolector de Actividades de Carrocería del Vehículo (MRAC)

- ✓ Para la realización de este módulo del proyecto lo primero que se ha de hacer es investigar sobre sensores que existen en los vehículos y hacer un estudio sobre la información que proporcionan.
- ✓ Una vez se decidan de qué sensores permiten obtener información interesante acerca de las interacciones del usuario con la carrocería del vehículo, habrá que estudiar de que manera se podrán leer los datos de tales sensores. Para ello habrá que investigar sobre protocolos que nos permitan conectarnos a la centralita de un vehículo y leer sus datos.
- ✓ Se estudiará alguna solución ya existente que permita leer los datos de los sensores de un vehículo a través de algún dispositivo externo, intentando hacer un estudio sobre alguna marca de vehículos.
- ✓ Una vez clasificada la información procedente del vehículo, que debe ser procesada, habrá que diseñar una jerarquía de eventos sensorizados que representen a los sensores de interés.

## 5.2 Módulo de Introducción de Itinerario por Usuario (MIIU)

- ✓ Para este módulo habrá que diseñar diversas estructuras de datos para representar los distintos conceptos de los que hace uso la aplicación, tales como tramo, su ocupación asociada o itinerario.
- ✓ Se deberá también proporcionar una herramienta para que el usuario sea capaz de introducir en el sistema los datos para definir las anteriores estructuras.

## 5.3 Módulo del Sistema de Almacenamiento de Itinerarios y Tramos (MSAITE)

- ✓ Una vez definidas las estructuras de datos que representan los conceptos de la sección anterior, se deberá especificar un mecanismo para la persistencia de los mismos. Dicho mecanismo deberá permitir almacenar y gestionar dichos datos para sobre ellos poder llevar a cabo tareas de adaptación supervisada off-line.
- ✓ Además este módulo deberá aplicar el mismo mecanismo de persistencia definido en el punto anterior para almacenar el flujo de actividades de carrocería tal como llegan al MIIU, las cuales servirán como entrada al módulo MGAI cuando la aplicación se ejecute en modo off-line.
- ✓ Tecnología necesaria: XML. Se ha de almacenar la información que va a etiquetar el usuario para que pueda ser gestionada más tarde, esto conlleva la serialización

de estos datos. La serialización es el proceso por el que un objeto cualquiera se puede convertir en una secuencia de bytes con la que más tarde se podrá reconstruir el valor de sus atributos. Más tarde, el proceso para deshacer la serialización volviendo a crear el objeto mediante la lectura de su estado a partir del almacenamiento es la deserialización. El mecanismo de persistencia serializará las estructura de datos de los tramos e itinerarios a través de ficheros XML [12], [13].

## 5.4 Módulo de Generación Automática Itinerario (MGAI)

- ✓ Para lograr la consecución de este módulo, se deberá diseñar una serie de patrones que permitan inferir los tramos e itinerarios realizados por el usuario, a partir de los datos recogidos e la carrocería del coche.
- ✓ Tecnología necesaria: CEP. El flujo de datos generado por los sensores del vehículo generan una gran cantidad de eventos diferentes, que deben ser monitorizados, filtrados, tratados, procesados, etc. para darles un sentido. Concretamente, el poder determinar el inicio y final de un tramo, los distintos tramos que forman un itinerario, así como la ocupación del vehículo en cada tramo. Todo este cometido casa perfectamente con la filosofía de la tecnología CEP [10], [14] y [15]. La aplicación de esta tecnología al presente trabajo se describe más en profundidad en la sección 6.4.

## 5.5 Módulo de Modelado de Usuario y Consolidación (MMUC)

- ✓ Para llevar a cabo los objetivos que se pretenden conseguir en este módulo habrá que diseñar un mecanismo de modelado on-line para, por un lado sostener los patrones diseñados y por otro modelar al usuario en cuanto a sus hábitos.
- ✓ Enfoque utilizado: Aprendizaje por refuerzo. Se realizará un estudio del enfoque del aprendizaje por refuerzo para ver en qué medida puede proporcionar una solución para la consecución del objetivo de este módulo. El aprendizaje por refuerzo sigue un esquema en el que un agente computacional opera en un entorno dinámico y, a través de una técnica de ensayo-error, aprende cuales son las acciones más apropiadas a realizar en cada momento.

## 5.6 Módulo de Sistema de Almacenamiento de Perfiles de Usuario (MSAPU)

- ✓ Diseñar una estructura de datos que defina el perfil de un usuario.
- ✓ Diseñar un mecanismo de persistencia que permita almacenar y recuperar dicha estructura de datos.

# 6 Diseño y Resolución

A partir de la Figura 4 se puede apreciar que cada una de las funcionalidades del prototipo diseñado en el presente trabajo pude verse como un módulo individual que interacciona con el resto. Por lo tanto, cada módulo ha sido diseñado teniendo en cuenta las tareas que abarca, así como sus interacciones con el resto de módulos de la arquitectura. A continuación se pasa a describir cada uno de ellos.

## *6.1 MRAC*

Este módulo se encarga de desarrollar las tareas acerca de las actividades de carrocería de un vehículo. Por tanto, en primer lugar es necesario hacer un pequeño estudio acerca de qué elementos de sensorización contiene hoy en día un automóvil y de qué manera la información que de ellos se obtiene puede ser útil para el presente proyecto.

### 6.1.1 Estado del arte de los Sensores del Automóvil

Los automóviles actuales tienen una cantidad importante de sensores (de 60 a 70 tipos de sensores diferentes en algunos modelos de vehículos). Estos sensores son necesarios para la gestión electrónica del automóvil y son utilizados por las centralitas electrónicas que gestionan el funcionamiento del motor, así como la seguridad y el confort del vehículo. En los últimos años, de entre todos los sensores de un automóvil, han cobrado mucha importancia los destinados a la seguridad.

A diferencia de los sensores convencionales, los utilizados en el sector del automóvil están diseñados teniendo en cuenta una serie de factores como son la alta fiabilidad, los bajos costes de producción y la alta precisión.

De entre la multitud de sensores existentes, a continuación se muestra un listado de los que a nuestro juicio se consideran los principales enfocados a la seguridad y al confort [16].

- ✓ Sensor de puertas. Informan de las puertas abiertas en cada momento. Disparan una alerta en caso de que el coche esté circulando y haya alguna puerta mal cerrada.
- ✓ Sensor de peso o de presencia: Indica los asientos que están ocupados. Hacen saltar alertas en caso de que se detecte presencia estando el coche cerrado [3], [4].

- ✓ Sensor de cinturones de seguridad. Detecta los cinturones que hay anclados. Dispara alerta en caso de que se detecte presencia sin cinturón anclado.
- ✓ Airbag. Informa de su funcionamiento.
- ✓ Sensores de iluminación. Permiten a la centralita informar de las distintas luces que hay activas en el vehículo en un momento determinado.
- ✓ Sensor luz automático: Se encienden las luces solas cuando detecta poca intensidad de luz en el exterior.
- ✓ ABS. Evita que las ruedas se bloqueen y el conductor pueda mantener el control del vehículo.
- ✓ Control de velocidad de crucero: Mantiene una velocidad determinada sin el uso permanente del acelerador
- ✓ Limpiaparabrisas inteligente. Enciende los limpiaparabrisas en caso de detectar agua en el exterior.
- ✓ Control de temperatura y climatización
- ✓ GPS: Sistema de posicionamiento
- ✓ Somnolencia: Mide la frecuencia del parpado del conductor y hace saltar alarma en caso de detectar indicios de sueño.
- ✓ Sensor que hace lectura de las señales de tráfico e informa al conductor si se excede del límite de velocidad establecido.

Con todo esto se ve que sería viable el tener una idea bastante acertada del comportamiento del conductor dentro del automóvil mediante la monitorización de los valores de los diferentes sensores del vehículo

Para que los datos de los sensores puedan ser procesados e interpretados necesita definirse en un formato para representar los datos que devuelven y un mecanismo para la recuperación y lectura de tal información. El principal sistema que permite hacer lectura de los datos de la centralita del coche es OBD II que se pasa a describir a continuación.

### 6.1.2 Protocolo de Diagnóstico a Bordo II (OBD II)

Las siglas ODB II se corresponden con “On Board Diagnostic Second Generation” [17], un protocolo estandarizado de codificación única que aporta un control casi

completo del motor y otros dispositivos del vehículo. Actualmente se encuentra implantado en casi todos los turismos y vehículos industriales ligeros que se producen.

### 6.1.2.1 Historia

Durante los años 70 y principios de los 80 del siglo pasado algunos fabricantes de coches empezaron a usar componentes electrónicos para el control y el diagnóstico de errores en sus automóviles. Al principio fue solo para conocer y controlar las emisiones de $CO_2$ del vehículo y adaptarlas a los estándares exigidos. Fue la Comisión de Recursos del Aire de California (CARB, California Air Resources Board) la que comenzó la regulación del protocolo OBD (On Board Diagnostic) para los vehículos vendidos en California.

La primera norma implantada fue la OBD I en 1988, donde se monitorizaban los parámetros de algunas partes del sistema del vehículo.

A partir de 1994, tanto la CARB como la Agencia de Protección del Medio Ambiente Estadounidense (EPA, Environmental Protection Agency) aumentaron los requerimientos del protocolo OBD, convirtiéndolo en el hoy conocido OBD II. Desembocando que a partir de 1996 los sistemas de sensores de los vehículos americanos tendrían que implementar dicho protocolo. Para vehículos europeos con motor de gasolina se implanta a partir del año 2000 y en el año 2003 para motores diesel.

### 6.1.2.2 Funcionamiento

En el protocolo ODB II la información de los sensores es procesada por la Unidades Electrónicas de Control (ECU, Electronic Control Unit) para comprobar si existe algún fallo en el vehículo. Los sensores permiten conocer a las ECU, cuáles son las condiciones externas (temperatura, humedad, niveles de aceite, etc.), y decidir cómo actuar en consecuencia sobre el motor. En caso de que alguno de los valores devueltos por los sensores se salga de los rangos marcados, el protocolo OBD II es el encargado de almacenar dicho suceso en forma de código de falla (DTC, Diagnostic Trouble Code) [18] y avisar al conductor mediante la señalización de un indicador luminoso o una señal acústica en el panel de control. El protocolo OBD II permite también leer los códigos de falla con el fin de hacer diagnósticos por parte de personal especializado.

Por tanto, no es el protocolo OBD II el encargado de detectar errores en el vehículo, sino, simplemente es el encargado de informar y almacenar el suceso de que un sensor está devolviendo valores anómalos y facilitar el acceso a esos valores.

OBD II también permite la conexión al CAN-BUS del vehículo, y así acceder al software de control a fin de monitorizar la red de sensores del vehículo. Esta última característica es extremadamente interesante para el presente proyecto ya que demuestra que se puede acceder a los valores devueltos por los sensores del coche a través del un dispositivo externo al vehículo en tiempo real.

En este sentido se ha tenido acceso a un lector OBD II, el *Workshop Information System (WIS)* [19] elaborado por Saab Automobile AB. Las pruebas realizadas con ese sistema han permitido descubrir gran cantidad de información del vehículo que se puede obtener a partir de los sensores.

#### 6.1.2.3 Códigos de Falla y su Coste

Como se explica en la sección anterior, ante unos valores anómalos en un sensor del vehículo, OBD II almacena tal suceso en forma de código de falla, de manera que más tarde se pueda obtener a partir de él información suficiente sobre su origen y causa.

Un código de falla se define por 5 dígitos en el cual cada dígito representa un valor diferente. El formato de los códigos sigue el patrón YXXXX, donde Y indica qué tipo de error es, pudiendo ser éste un error del motor, transmisión, carrocería o chasis. El segundo dígito indica la organización responsable de definir el código, pudiendo tomar el valor 0 lo que representa que es un error común a todas las marcas, o el valor 1 que se reserva a los distintos fabricantes del vehículo. El tercer dígito representa la función específica del vehículo (sistema electrónico, control de aire y combustible, control de velocidad, entradas y salidas de las ECU, etc.). Los dos últimos dígitos están relacionados específicamente con la falla o error.

Como se ha mencionado, algunos de los códigos son reservados para uso de los fabricantes, en concreto, los códigos que recogen información de la carrocería y confort, conocidos también como *Body Codes*. Precisamente éstos son los códigos a los que sería interesante acceder en este proyecto. Desafortunadamente, al estar reservados estos códigos para cada marca, tienen su propia encriptación de datos y no se hacen públicos.

Una de las principales instituciones que proporcionan los códigos propietarios de los diferentes fabricantes de coches es la Equipment and Tools Institute (ETI) [20], una asociación comercial formada por empresas de todos los segmentos de la industria del automóvil, teniendo más de 70 compañías miembro.

Dicha asociación cuenta con la ETI's TEK-NET Library, una base de datos tecnológica y de documentación con todos los códigos de los distintos vehículos. El acceso a dicha base de datos está sólo permitido a los miembros de la ETI. Sin embargo, hacerse miembro de este instituto tiene un coste de 7.500 $ anuales.

Los datos técnicos de la ETI's TEK-NET Library cubren un gran mercado de vehículos pero no incluye a los vehículos del fabricante General Motors (GM). Los códigos de dicha empresa están solo accesibles para sus socios privados. Por tanto, para poder acceder a sus códigos propietarios, que permiten interpretar los datos de las centralitas de sus vehículos, habría que hacerse socio privado de la empresa lo cual exige un pago aproximado de 50.000 $.

### 6.1.3 Viabilidad del MRAC

Como se acaba de explicar en la sección anterior, para conseguir los Body Codes de una centralita de coche y poder así acceder a los valores que nos proporcionan los sensores acerca de las actividades de carrocería que ocurren en el vehículo, se debería hacer un gasto bastante importante de dinero.

Por tanto, la viabilidad para el desarrollo de este módulo del presente proyecto, está sujeta a un gran desembolso económico para obtener los códigos contenidos tanto en la ETI's TEK-NET Library como los propietarios de GM.

Si en un futuro se pudieran conseguir los códigos de algún fabricante, únicamente habría que desarrollar un adaptador que transforme la salida real del automóvil (en formato OBD II) a la entrada esperada por el sistema que en este proyecto se desarrolla.

Llegados a este punto y no habiéndose podido conseguir los códigos reales de ningún fabricante, con lo que no se podrá utilizar un vehículo de verdad para el desarrollo del sistema, se decidió adoptar como solución la implementación de un *simulador de actividades de carrocería* (SAC) junto con un *filtro de sensorización* (FS). Mientras el SAC va a permitir introducir información de las diferentes actividades que un usuario realiza dentro de su coche, el FS transformará algunas de dichas actividades en datos recogidos por los sensores.

### 6.1.4 Simulador de Actividades de Carrocería (SAC)

El esquema inicial del SAC se muestra en la Figura 5:

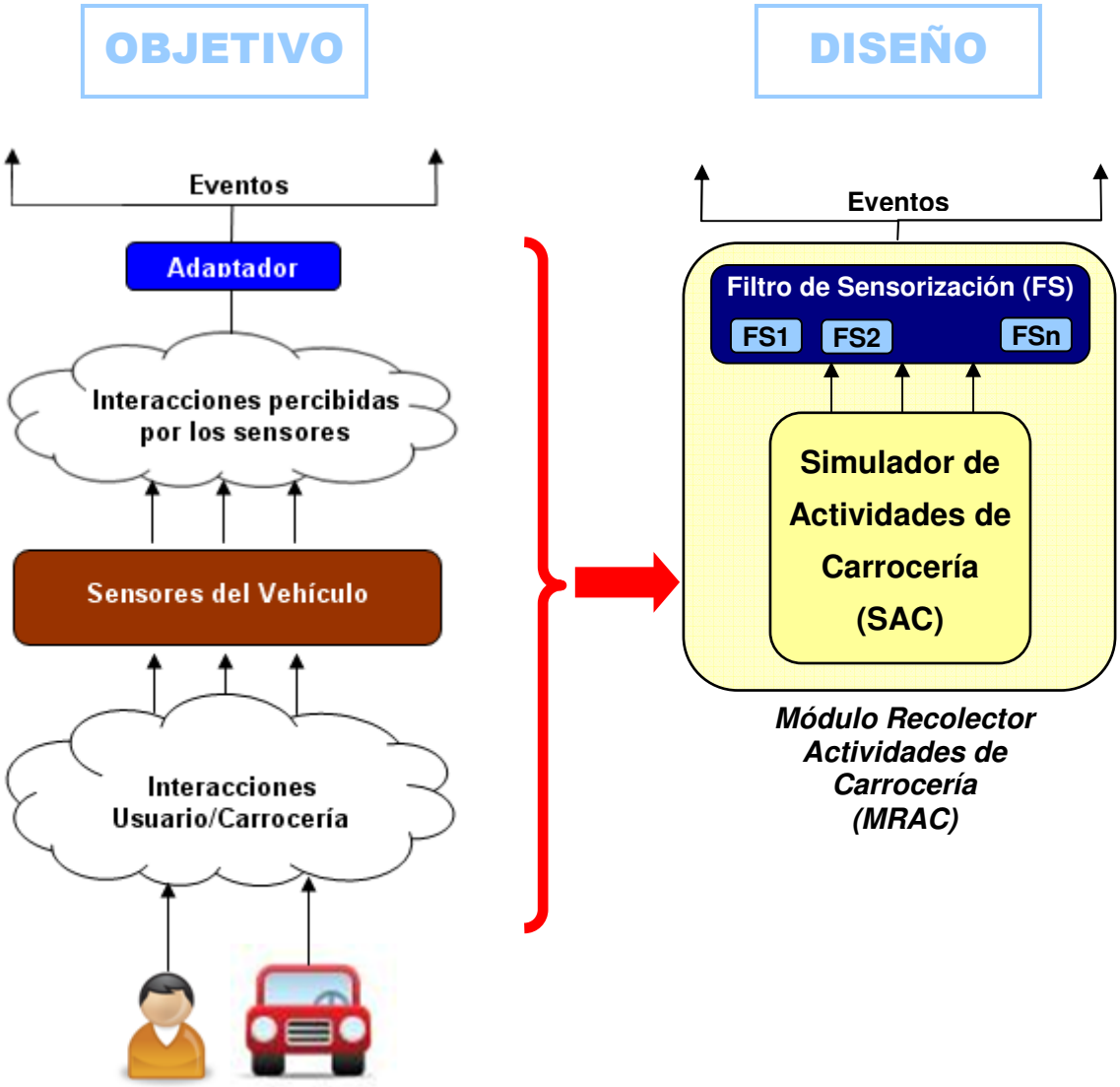



Figura 5. Diseño del MRAC

Desde el SAC se podrán simular muchas de las actividades que un humano realiza en un automóvil. Así, deberá obligar a seguir un orden lógico en las actividades como por ejemplo, no permitir bajar una ventanilla si todavía no se ha abierto el coche ni hay nadie dentro o no poder acelerar si el motor no está todavía encendido.

Se muestra a continuación un listado de todas las actividades que un usuario va a poder realizar a través del simulador. Los nombres son significativos del cometido que tienen.

***Cerraduras***

- Abrir/Cerrar cerradura conductor
- Abrir/Cerrar cerradura copiloto
- Abrir/Cerrar cerradura trasera dcha
- Abrir/Cerrar cerradura trasera izq
- Abrir/Cerrar puerta conductor
- Abrir/Cerrar puerta copiloto
- Abrir/Cerrar puerta trasera dcha
- Abrir/Cerrar puerta trasera izq
- Abrir/Cerrar maletero
- Abrir/Cerrar capo

***Elevalunas***

- Subir/Bajar elevalunas copiloto
- Subir/Bajar elevalunas conductor
- Subir/Bajar elevalunas trasero dcho
- Subir/Bajar elevalunas trasero izq
- Abrir/Cerrar techo solar
- Bloquear/Desbloquear elevalunas

***Seguridad***

- Poner/Quitar cinturón conductor
- Poner/Quitar cinturón copiloto
- Poner/Quitar cinturón trasero central
- Poner/Quitar cinturón trasero dcho
- Poner/Quitar cinturón trasero izq
- Activar/Desactivar airbag delanteros
- Activar/Desactivar airbag laterales

***Iluminación***

- Encender/Apagar intermitente izq
- Encender/Apagar intermitente dcho
- Encender/Apagar cuatro intermitentes
- Encender/Apagar faro cruce
- Encender/Apagar faro carretera
- Encender/Apagar antiniebla delantero
- Encender/Apagar antiniebla trasero
- Encender/Apagar luz remolque
- Encender/Apagar luz freno
- Encender/Apagar luz marcha atrás
- Encender/Apagar luz maletero
- Encender/Apagar luces interruptores
- Encender/Apagar luces compartimento

***Información***

- Meter/Sacar llave
- Arrancar/Parar motor
- Marcha de la caja de cambios
- Velocidad automóvil
- Número de revoluciones
- Angulo de giro
- Angulo de volante
- Activar/Desactivar pedal freno
- Tocar bocina
- ABS activado/desactivado
- Indicador combustible bajo
- Encender/Apagar sensor lluvia
- Encender/Apagar sensor luz
- DatosGPS
- Activar/Desactivar velocidad crucero
- Disparada/Apagada alarma antirrobo
- Disparado/Apagado sensor sueño
- Encender/Apagar sensor lector señales

***Equipo Sonido***

- Encender/Apagar radio
- Encender/Apagar lector de discos

***Limpiadores***

- Encender/Apagar parabrisas delantero
- Encender/Apagar parabrisas trasero
- Encender/Apagar lavafaros

***Calefacción AC***

- Temperatura interior
- Temperatura exterior
- Encender/Apagar aire acondicionado
- Encender/Apagar velocidad ventilador
- Encender/Apagar calefacción trasera
- Encender/Apagar recirculación aire

***Retrovisores***

- Mover retrovisor conductor
- Mover retrovisor copiloto
- Ajustar Memoria

***Asiento***

- Ocupar/Desocupar asiento conductor
- Ocupar/Desocupar asiento copiloto

| | |
|---|---|
| Ocupar/Desocupar asiento tras izq | Poner/Quitar silla niño tras izq |
| Ocupar/Desocupar asiento tras central | Poner/Quitar silla niño tras central |
| Ocupar/Desocupar asiento tras dcho | Poner/Quitar silla niño tras dcho |
| Poner/Quitar silla niño copiloto | |

Las actividades de carrocería identifican actividades humanas que ocurren en el vehículo y cada una tendrá distintos datos según su propia naturaleza. Así, por ejemplo la actividad que recoge si una puerta está abierta tendrá un atributo que indique abierta o cerrada, mientras que la actividad que recoge la velocidad tendrá un atributo que indique los km/h. Aún así todas tienen un atributo común que indica el momento exacto en el que ocurre la actividad.

### 6.1.5 Filtro de Sensorización (FS)

No todos los coches tienen los mismos sensores ya que, por ejemplo, algunos modelos de vehículos son capaces de detectar cuando nos ponemos el cinturón de seguridad y otros no. Por lo tanto, algunas de las actividades generadas por el SAC tendrán su representación en datos recogidos por los sensores pero otras no. Dicho mapeo entre actividad del mundo real y datos reflejados por los sensores es realizado por el FS.

Por este motivo, habría que desarrollar un filtro u otro en función del perfil del coche que se quiera simular. Un coche totalmente sensorizado, de gama alta, se modelaría con un FS que transformaría todas las actividades generadas por el simulador de actividades en datos recogidos por los sensores. Un coche con una sensorización estándar de gama media se representaría mediante un FS que transformaría sólo algunas de las actividades, mientras un coche con poca sensorización tendría un filtro que transformaría solo unas pocas. Se observa en la Figura 5 que se pueden tener varios FS y según el tipo de coche que se quiera simular los datos del SAC pasarán por uno u otro.

En este sentido hay que señalar que el FS que se ha desarrollado para el presente trabajo es el equivalente a un coche con perfil de gama media, concretamente las actividades que han sido transformadas en datos de los sensores se han basado en los sensores y alarmas de un *Citroën C4 Collection*.

Concretamente, las actividades detectadas por el FS del presente trabajo son las que aparecen marcadas (✓) en el siguiente listado:

***Cerraduras***

- Abrir/Cerrar cerradura conductor ✓
- Abrir/Cerrar cerradura copiloto ✓
- Abrir/Cerrar cerradura trasera dcha ✓
- Abrir/Cerrar cerradura trasera izq ✓
- Abrir/Cerrar puerta conductor ✓
- Abrir/Cerrar puerta copiloto ✓
- Abrir/Cerrar puerta trasera dcha ✓
- Abrir/Cerrar puerta trasera izq ✓
- Abrir/Cerrar maletero ✓
- Abrir/Cerrar capo ✓

***Elevalunas***

- Subir/Bajar elevalunas copiloto ✓
- Subir/Bajar elevalunas conductor ✓
- Subir/Bajar elevalunas trasero dcho ✓
- Subir/Bajar elevalunas trasero izq ✓
- Abrir/Cerrar techo solar ✓
- Bloquear/Desbloquear elevalunas ✓

***Seguridad***

- Poner/Quitar cinturón conductor ✓
- Poner/Quitar cinturón copiloto ✓
- Poner/Quitar cinturón trasero central ✓
- Poner/Quitar cinturón trasero dcho ✓
- Poner/Quitar cinturón trasero izq ✓
- Activar/Desactivar airbag delanteros ✓
- Activar/Desactivar airbag laterales ✓

***Iluminación***

- Encender/Apagar intermitente izq ✓
- Encender/Apagar intermitente dcho ✓
- Encender/Apagar luces emergencia ✓
- Encender/Apagar faro cruce ✓
- Encender/Apagar faro carretera ✓
- Encender/Apagar antiniebla ✓
- Encender/Apagar antiniebla trasero ✓
- Encender/Apagar luz remolque ✓
- Encender/Apagar luz freno ✓
- Encender/Apagar luz marcha atrás ✓
- Encender/Apagar luz maletero ✓
- Encender/Apagar luces interruptores ✓
- Encender/Apagar luz compartimento ✓

***Información***

- Meter/Sacar llave ✓
- Arrancar/Parar motor ✓
- Marcha de la caja de cambios ✓
- Velocidad automóvil ✓
- Número de revoluciones ✓
- Angulo de giro ✓
- Angulo de volante ✓
- Activar/Desactivar pedal freno ✓
- Tocar bocina ✓
- ABS activado/desactivado ✓
- Indicador combustible bajo ✓
- Encender/Apagar sensor lluvia ✓
- Encender/Apagar sensor luz ✓
- DatosGPS
- Activar/Desactivar velocidad crucero
- Disparada/Apagada alarma antirrobo
- Disparado/Apagado sensor sueño
- Encender/Apagar sensor lector señales

***Equipo Sonido***

- Encender/Apagar radio ✓
- Encender/Apagar lector de discos ✓

***Limpiadores***

- Encender/Apagar parabrisas ✓
- Encender/Apagar parabrisas trasero ✓
- Encender/Apagar lavafaros

***Calefacción AC***

- Temperatura interior ✓
- Temperatura exterior ✓
- Encender/Apagar aire acondicionado ✓
- Regular velocidad ventilador ✓

Encender/Apagar calefacción trasera ✓

Encender/Apagar recirculación aire ✓

***Retrovisores***

Mover retrovisor conductor ✓

Mover retrovisor copiloto ✓

Ajustar Memoria ✓

***Asiento***

Ocupar/Desocupar asiento conductor

Ocupar/Desocupar asiento copiloto

Ocupar/Desocupar asiento tras izq

Ocupar/Desocupar asiento tras central

Ocupar/Desocupar asiento tras dcho

Poner/Quitar silla niño copiloto

Poner/Quitar silla niño tras izq

Poner/Quitar silla niño tras central

Poner/Quitar silla niño tras dcho

Se puede observar que aparecen muchas actividades como marcadas, esto es porque, aunque se haya simulado un coche de gama media, hoy día los nuevos vehículos incorporan numerosos sensores de serie. Sin embargo, el FS desarrollado no es capaz de detectar los ocupantes del vehículo al no tener sensores de peso en los asientos.

Hay que recordar que si se hubieran tenido acceso a los códigos propietarios de alguna marca de vehículos no habría hecho falta desarrollar ni SAC ni el FS, ya que el prototipo podría haberse probado directamente en un vehículo real.

### 6.1.6 Flujos de trabajo generados

Hasta este punto, se ha descrito el funcionamiento del módulo en el cual, de la manera descrita anteriormente, se va a producir un flujo de actividades de carrocería que el usuario va a ser capaz de generar a partir del SAC.

Una vez el usuario ha introducido las actividades en el simulador, ésas van a parar a dos módulos diferentes:

1) MIIU. De esta forma, el usuario va a poder agrupar las actividades en tramos, itinerarios e indicar la ocupación del vehículo en cada tramo, tal como se verá en el siguiente apartado.

2) MGIA. Las actividades, pasando por el FS, se convertirán en datos que se procesarán en dicho módulo, para que infiera por sí solo tanto los diferentes tramos del itinerario como la ocupación del vehículo en cada tramo.

### 6.1.7 Modos de funcionamiento

Así, también hay que destacar que el MRAC puede funcionar de dos modos distintos:

1) On-line: Conforme el usuario va generando las actividades en el SAC, estas van pasando por el filtro de sensorización y entrado directamente al MGIA.

2) Off-line: Todas las actividades generadas en el simulador se almacenan en un fichero XML a través del MSAITE. Esto permite que más tarde se pueda arrancar la aplicación y seleccionar el fichero de actividades sobre el que se quiere trabajar. El sistema cargará todas las actividades del fichero XML y las enviará al FS para que a partir de ahí siga la línea de ejecución del sistema. De esta forma, no es necesario que un usuario esté en ese momento introduciendo actividades en el SAC.

## *6.2 MIIU*

Se han mencionado en el presente trabajo, en repetidas ocasiones, los términos de *itinerario* y *tramo*. Esta sección se basa principalmente en estos dos conceptos que se definen a continuación.

Se va a considerar una unidad de información llamada itinerario para describir todo lo que hace un usuario con su vehículo desde que sale de su punto origen hasta que regresa a él. Si se busca la definición de itinerario en el diccionario de la Real Academia Española (RAE) se presenta la siguiente definición: “Dirección y descripción de un camino con expresión de los lugares, accidentes, paradas, etc., que existen a lo largo de él. Guía, lista de datos referentes a un viaje.”. Adaptando esta definición ámbito del presente trabajo, se entenderá como un itinerario toda la ruta que un usuario realizará con su vehículo hasta regresar al punto origen, con lo que se va a considerar una ruta circular. Si se quiere definir el itinerario de un viaje, se entenderá como tal la ruta completa hasta el regreso al punto origen aunque esto suponga que el regreso ocurra días después de la salida.

Se entenderá que un itinerario está dividido en *tramos*. Según la RAE un tramo es: “Cada una de las partes en que está dividido o se puede dividir un camino”. Esta definición concuerda perfectamente con el uso que se va a hacer del término en la aplicación. Como se describió anteriormente, un itinerario tendrá forma circular ya que su origen y su destino son el mismo punto y las secciones en las que se puede dividir se denominarán tramos. La unión de todos los tramos creará la ruta completa del

Itinerario. Cada tramo tiene un punto origen y destino diferentes aunque el punto de origen de un tramo coincide siempre con el final del anterior. Un gráfico de dicho itinerario se muestra en la Figura 1.a).

Un tramo quedará identificado a parte de por sus puntos tanto espaciales como temporales de inicio y fin, por la ocupación del vehículo a lo largo de todo el tramo. Se muestra a continuación un ejemplo ilustrativo; Un usuario que cada *lunes* sale de su casa sobre las 9 de la mañana, en este momento la ocupación del habitáculo está compuesta por un adulto en el sitio del conductor y por un niño en el asiento trasero izquierdo. El primer tramo que realiza va desde su casa hasta el colegio del niño. En este momento cambia la ocupación del habitáculo y comienza un nuevo tramo en el que se dirige únicamente el conductor hacía su sitio de trabajo. Al medio día, el usuario sale del trabajo y se dirige a comer a casa de su madre, identificándose así otro nuevo tramo. Finalmente regresa a su casa junto con otro adulto en el sitio del copiloto y el niño atrás. Al regresar a casa se cierra el Itinerario que como se ha visto está compuesto cuatro tramos. El punto de origen de cada tramo coincide siempre con el final del anterior.

Los diferentes datos generados por el MRAC son posteriormente procesados por el MIIU, donde el usuario realizará un “etiquetado” de las actividades agrupándolas en las unidades de información descritas anteriormente. Por tanto, el usuario a través de este módulo debe realizar tres tareas a partir de los datos de entrada:

a) Identificar itinerarios

b) Identificar tramos

c) Identificar la ocupación del vehículo para cada tramo.

Es necesario para el sistema que el usuario introduzca estos términos ya que el MGAI intentará inferir itinerarios a partir de las mismas actividades. De esta forma se podrá comparar si el itinerario inferido coincide con el que el usuario ha indicado a través de este módulo.

Para que el usuario pueda agrupar correctamente los datos provenientes del MRAC en tramos e itinerarios deben de definirse formalmente ambos términos. Pasemos a continuación a definir dichos términos.

### 6.2.1 Itinerario

Como ya se ha dicho, un itinerario se va a considerar, todo lo que hace el coche desde que sale de un punto de origen (la casa del propietario, un parking, etc.), hasta que vuelve a el, es decir, todo el recorrido que hace un vehículo en un periodo de tiempo hasta que regresa al mismo punto del que partió inicialmente. Para la aplicación, se ha diseñado el concepto itinerario definida mediante la siguiente tupla:

`IT: {Id, idUsuario, nTramos, secTramos}`, donde:

- Id: Identificación del Itinerario, nombre descriptivo
- idUsuario: DNI y nombre del usuario del itinerario
- nTramos: Número de tramos que componen el itinerario
- secTramos: Secuencia de Tramos. Esta será una estructura que almacena tantas instancias de Tramo, como número de tramos tenga el Itinerario.

### 6.2.2 Tramo

Como se ha descrito al principio de esta sección, los tramos se consideran como las distintas secciones en las que se puede dividir un itinerario. Un tramo se caracteriza por tener desde el origen al destino los mismos ocupantes y carga. En caso de que hubiera un cambio de ocupación, es decir, se para el coche para que alguien se baje o suba de el, este punto de inflexión provocaría el final de un tramo e inicio del siguiente.

Para la aplicación, se ha diseñado el concepto *tramo*, de manera que estará compuesto por la siguiente tupla:

`Tr: {id, PES, PTS, PTF, ocupantes, secActividades}`, donde:

- id: Identificación del tramo, nombre descriptivo.
- PES: Punto espacial de origen (desde dónde sale).
- PTS: Punto temporal de origen (a qué hora sale).
- PTF: Punto temporal de destino (a qué hora llega).
- Ocupantes: Se define cuantos ocupantes hay y donde están situados. La estructura y metodología de esta estructura se verá a continuación.

- secActividades: Secuencia de actividades de carrocería que ocurren durante el tramo.

Los puntos espaciales se definen a partir de los siguientes datos:

`PE: {exacto, longitud, latutid}`, donde:

- exacto: Indica si punto de partida es exacto o en cambio puede tomar cualquier valor en una zona. Si siempre se aparca en el mismo sitio, como una plaza de garaje, las coordenadas GPS siempre serán las mismas, en cambio, si hay que buscar aparcamiento por una zona, el punto de partida tomará coordenadas GPS distintas cada vez, estando estas acotadas en un perímetro.
- Longitud: Coordenada GPS.
- Latitud: Coordenada GPS.

Los puntos temporales se definen a partir de los siguientes datos:

`PT: {exacto, fecha, hora}`, donde:

- exacto: Indica si el instante es exacto o aproximado. Si se indica una hora, con la marca de exacto, por ejemplo las 9 de la mañana se estará diciendo que a las 9 en punto es cuando está ocurriendo el inicio o final de tramo. En cambio, si estamos indicando rutinas en las que se marcan las 9 de la mañana sin la marca de exacto, se está dando un valor aproximado del punto temporal, pudiendo tomar distintos valores cercanos a la hora indicada.
- fecha: fecha.
- hora: hora.

El primer tramo indicará el origen espacial y temporal del itinerario. Se puede apreciar que en los tramos no se indica el punto espacial destino, ya que siempre coincidirá con el origen del siguiente tramo.

El último tramo del itinerario debe coincidir en el punto espacial con el primero, ya que como se ha dicho anteriormente un itinerario se completa cuando el coche vuelve al punto de origen. Puede tener tantos tramos como se quiera, si se contempla por ejemplo un viaje familiar, el itinerario será desde que salen de casa hasta la vuelta, y estará formado por múltiples tramos.

Un cambio de tramo se identifica por una de las siguientes casuísticas:

- Que varíe la ocupación (un tramo tiene siempre la misma ocupación de principio a fin)
- Que se apague el motor
- Que el coche permanezca al ralentí más de dos minutos

### 6.2.3 Ocupación del Habitáculo

Otra información de crucial importancia que el usuario debe introducir a través del MIIU es qué tipo de ocupantes existe en el interior del vehículo (llamado en adelante habitáculo) en cada tramo. En este sentido, se distinguen seis tipos de departamentos dentro del habitáculo:

- Conductor (C).
- Copiloto (CP).
- Trasero izquierdo (TI).
- Trasero central (TC).
- Trasero derecho (TD).
- Maletero (M).


Para un departamento concreto, debe existir una ocupación determinada. El diseño de los diferentes tipos de ocupación es un aspecto clave ya que influirá en gran medida en la utilidad del sistema.

Con el fin de modelar el tipo de ocupación que puede existir en cada departamento del habitáculo del vehículo se ha diseñado la taxonomía mostrada en la Figura 6.

Dicha taxonomía se compone de ocho términos diferenciados denominados concepto ocupación (COcupacion). Cada uno de los cuales se diferencia del resto tanto por su especificidad como por su fragilidad. A mayor especificidad, mayor es la cantidad de información que se proporciona acerca de la ocupación. Por ejemplo, a nivel de especificidad 1, se puede saber si un departamento está vacío u ocupado, sin saber exactamente qué o quién es lo que lo está ocupando. El nivel 2 ya aporta más información distinguiendo la ocupación entre una carga (que puede ser un bolso, maleta, trasportín de animales, etc.) o bien una persona. El nivel 3 determina si una persona es un adulto o un menor, al igual que el nivel 4 distingue si un menor puede

ser niño o bebe. Todos los términos de la misma rama tienen un mismo padre común en la taxonomía ya que, por ejemplo, tanto un adulto como un niño son ambos personas.

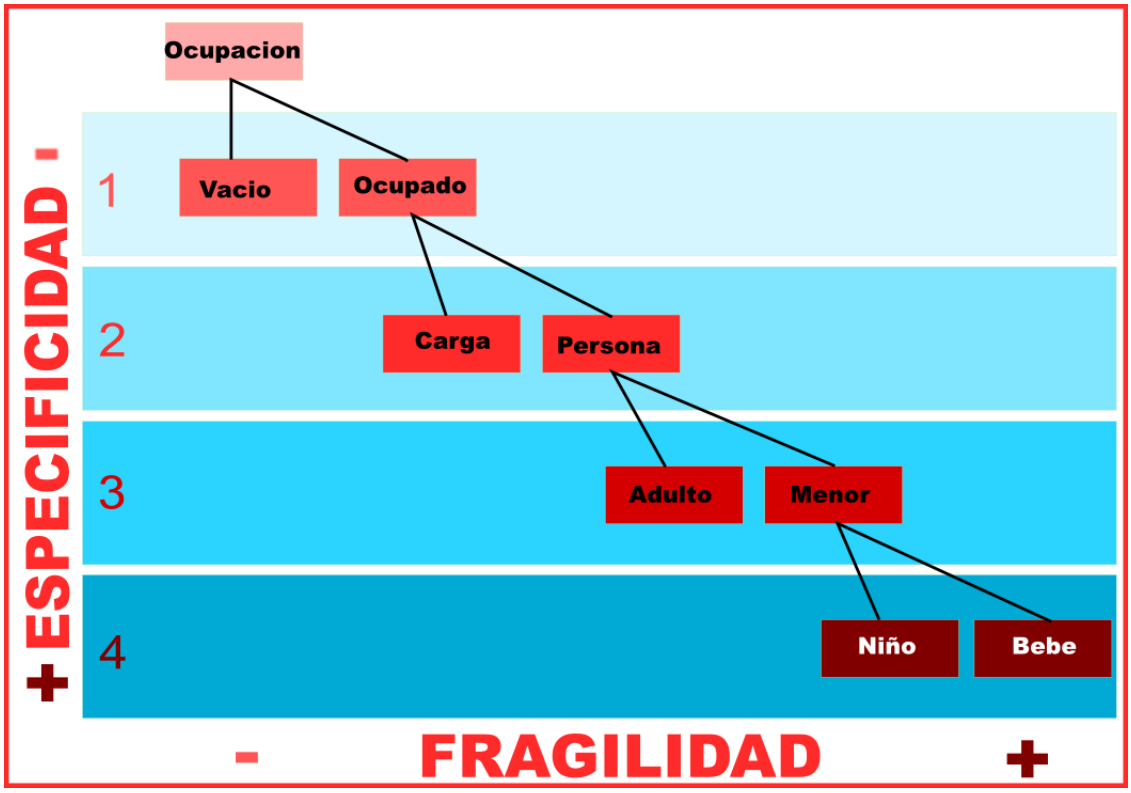



Figura 6. Taxonomía de Ocupación

En cada nivel de especificidad se observa un hijo izquierdo, que será siempre un nodo hoja, y otro derecho que se podrá desglosar en otros dos nodos en el siguiente nivel de especificidad, constituyendo así un árbol binario.

Cada término de la taxonomía tiene distinta fragilidad que aumenta de forma proporcional al nivel de especificidad. Así, habrá más fragilidad conforme el término se sitúe más a la derecha. Cuanta más fragilidad tiene un COcupacion, más vulnerable es a sufrir daños, ya que por ejemplo es evidente que un bebe deberá ser tenido más en cuenta que una carga.

#### 6.2.3.1 Concepto de Similitud Semántica-Jerárquica (SSJ)

Para representar la diferencia existente entre dos COcupacion diferentes, en [21] se define una unidad de medida de distancia entre dos nodos de un árbol denominada Similitud Semántica-Jerárquica (SSJ). Dicho concepto se basa en la diferencia de profundidad entre los dos nodos a comparar y en qué posición en árbol se encuentra

su nodo padre común. Además, esta medida toma un rango de valores que oscila entre 0 y 1, representado el valor 1 que ambos nodos son el mismo y el valor 0 que son totalmente diferentes.

A continuación se muestra el pseudocódigo que define de manera formal el concepto de SSJ adaptado a la taxonomía de COcupaciones presentada:

**Entrada:**

- $COcupacion_a^k$, $k \in$ {*C,CP,TI,TC,TD,M*}: Primer COcupacion tomado como entrada.

- $COcupacion_b^k$, $k \in$ {*C,CP,TI,TC,TD,M*}: Segundo COcupacion tomado como entrada.

**Salida:**

- $SSJ(\ COcupacion_a^k, COcupacion_b^k) \in$ {0..1}: Valor entre 0 y 1 que representa la distancia entre los dos conceptos ocupación de entrada.

**Funciones Auxiliares:**

- $ACC(\ COcupacion_a^k, COcupacion_b^k)$: Representa el antecesor común más cercano de $COcupacion_a^k$ y $COcupacion_b^k$. Calcula el nodo padre común más cercano entre dos COcupacion de la taxonomía de ocupación. Por ejemplo $ACC(Adulto, Bebe) = Persona$ y $ACC(Persona, Menor) = Persona$

- $NE[COcupacion^k]$: Devuelve el nivel de especificidad de $COcupacion^k$. Por ejemplo $NE[Menor] = 3$ y $NE[Niño] = 4$

- $MAX[entero_a, entero_b]$: Devuelve el máximo de los dos números enteros

**Fórmula:**

$$SSJ(\ COcupacion_a^k, COcupacion_b^k) = \frac{NE\left[ACC(COcupacion_a^k, COcupacion_b^k)\right]}{MAX\left(NE\left[COcupacion_a^k\right], NE\left[COcupacion_b^k\right]\right)}$$

Por ejemplo del SSJ entre los conceptos ocupación persona y adulto daría como resultado:

$$SSJ(Persona, Adulto) = \frac{NE[Persona]}{MAX(NE[Persona], NE[Adulto])} = \frac{2}{MAX(2,3)} = \frac{2}{3} = 0{,}66$$

La SSJ permite medir la "bondad" de las inferencias sobre la ocupación del habitáculo que está realizando el sistema. Así, si se dispusiese de la ocupación real de cada departamento, bastaría con obtener el SSJ entre el COcupacion inferido y el real para determinar cómo de buenas son las inferencias del sistema, teniendo en cuenta que valores próximos a 1 indican buenas inferencias y valores próximos a 0 inferencias erróneas.

### 6.2.4 Salida del módulo

Como resultado de la funcionalidad que el MIIU ofrece al usuario, las actividades introducidas en el simulador quedarán troceadas en tramos constituyéndose un árbol parecido al mostrado en la Figura 7.

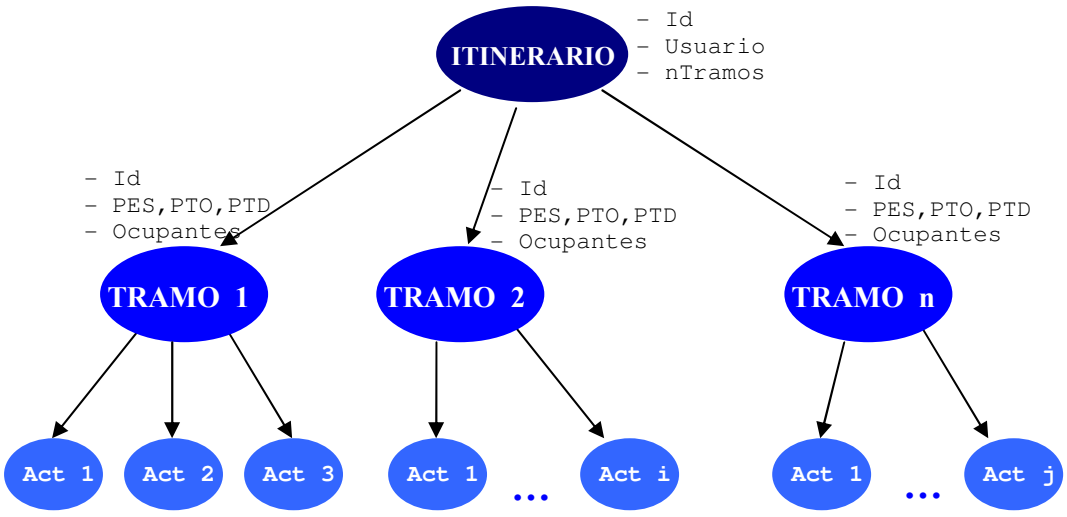



Figura 7. Estructura en árbol de un itinerario etiquetado por un usuario

## *6.3 MSAITE*

Los objetivos que persigue este módulo son tanto el almacenamiento y persistencia como la posterior recuperación de las actividades de carrocería sin etiquetar y también de los itinerarios generados por el usuario a través del MIIU a partir del etiquetado de estas mimas actividades de carrocería.

Tanto las actividades de carrocería como los itinerarios etiquetados son almacenados en un repositorio. Al estar trabajando con un prototipo que no será utilizado por una gran cantidad de usuario se decidió que dicho repositorio consistiera en un sistema de ficheros en vez de, por ejemplo una base de datos relacional que resulta más apropiada cuando se tiene una gran cantidad de datos. Por lo tanto, las actividades de carrocería y los itinerarios etiquetados serán almacenadas en ficheros diferentes.

## *6.4 MGAI*

Este es el módulo principal del sistema ya que es el encargado de recibir como entrada la gran cantidad de datos recogidos por los sensores de un automóvil (del FS en el caso del presente proyecto). Su objetivo principal es intentar obtener información útil a partir de sus datos de entrada, es decir, detectar lo que está haciendo el usuario dentro del vehículo. Además, los datos de entrada vienen en forma de patrones. Por ejemplo, si se recibe que se abre la puerta del conductor, a continuación se cierra y posteriormente el cinturón del conductor se ha anclado, se puede deducir que un usuario se ha sentado en el asiento del conductor. Todo esto casa perfectamente con la filosofía del enfoque de Procesamiento de Eventos Complejos (CEP, Complex Event Processing) que se describe a continuación.

### 6.4.1 Introducción al diseño de Procesamiento de Eventos Complejos

El objetivo principal cuando se utiliza CEP es detectar situaciones del mundo real, llamadas *actividades*. Como se aprecia en la Figura 8, se han de determinar qué acciones del mundo real se quieren detectar para su procesamiento en CEP. Tal como se muestra en la parte izquierda de la Figura 8, cada actividad de alto nivel se ha de ir descomponiendo en actividades de más bajo nivel hasta convertirlas en actividades básicas que tienen su traducción en el sistema de información (SI), denominadas en terminología CEP eventos.

Posteriormente, a partir de dichos eventos en el SI, se detectan nubes de relaciones entre los eventos simples y se van creando eventos más complejos como resultado de combinaciones de ellos hasta que, finalmente, tal como se muestra en la parte derecha de la Figura 8, se obtienen uno o varios eventos complejos que representan a la actividad del mundo real de la que se ha partido inicialmente.

Así por ejemplo, se puede entender como actividad del mundo real la de entrar en el coche, la cual se puede descomponer en actividades básicas como abrir la puerta,

sentarse en el asiento, cerrar la puerta y ponerse cinturón. Estas actividades básicas tienen su reflejo en la información aportada por los sensores del coche que se convierten en eventos a ser procesados por el sistema. Por otro lado, en el sistema se definen patrones para determinar la relación entre eventos básicos del tipo, si "abre puerta" y después "cierra puerta" se detecta que el departamento del coche se ha ocupado y si después "pone cinturón" el sistema detecta que una persona ha entrado en el coche mediante la generación de un evento complejo que tiene su correspondencia en una actividad del mundo real.

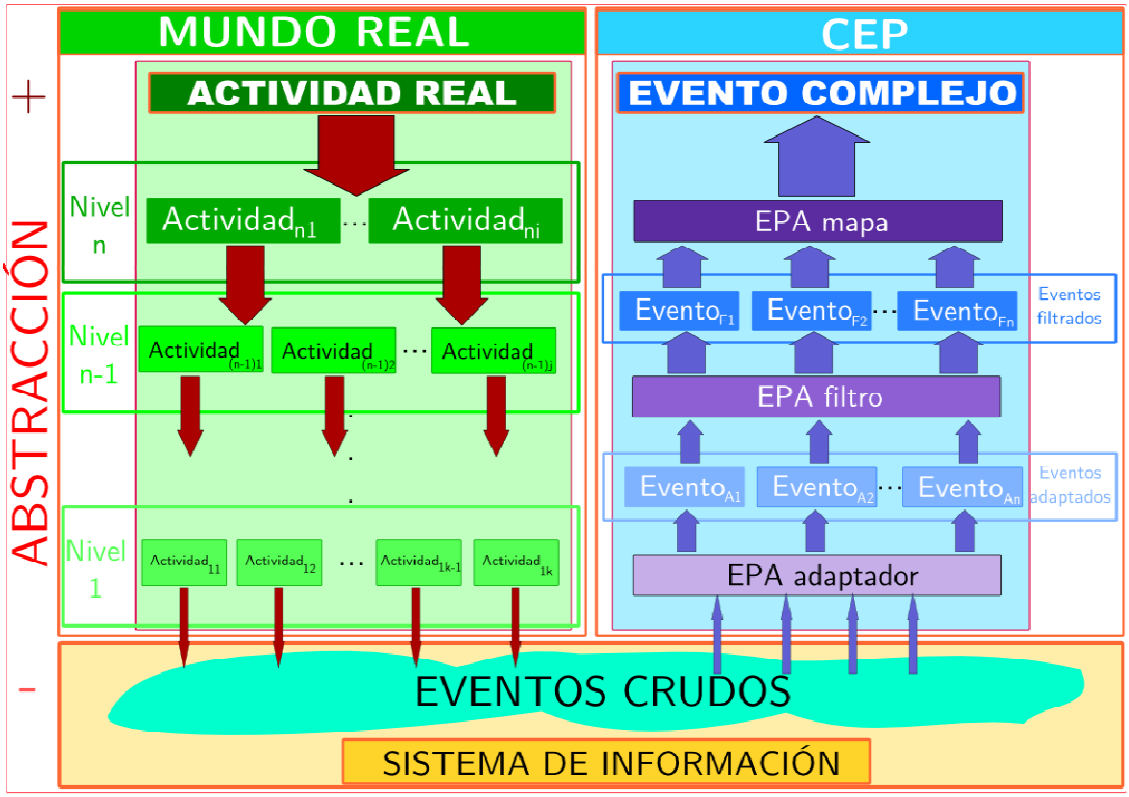



Figura 8. Esquema del procesamiento basado en eventos complejos

Las relaciones entre eventos se definen por medio de patrones. A partir de los patrones se construyen las denominadas Reglas de Procesamiento de Eventos (EPRs, Event Processing Rules) que asocian una determinada acción a realizar cada vez que se detecta un patrón en el flujo de eventos. Para generar tanto las EPRs como los patrones se utiliza un Lenguaje de Procesamiento de Eventos (EPL, Event Processing Language) [22].

Sobre la base de estas EPRs, se diseñan los Agentes de Procesamiento de Eventos (EPAs, Event Processing Agents). Un EPA contiene un *motor de procesamiento de eventos* sobre el que se ejecuta una serie de EPRs. A su vez, los

EPAs pueden conectarse entre sí formando jerarquías denominadas Redes de Procesamiento de Eventos (EPN, Event Processing Network).

La clave de CEP es la continua identificación de patrones complejos a partir del flujo de eventos que pasa a través de la EPN. En tiempo de ejecución van entrando al motor de cada EPA eventos heterogéneos procedentes de diversas fuentes (sensores, bases de datos, servicios web, etc.). El motor de cada EPA busca correspondencias entre los eventos de entrada y los patrones definidos en sus EPRs de forma que, cada vez que se active uno de ellos se realiza su acción asociada. Estas acciones pueden ser desde la generación de mensajes y alertas a la agregación o generación de eventos de nivel superior (reconocer un grupo de eventos de nivel inferior de entre todos los eventos y crear un evento que resuma al grupo anterior), para ser procesados por otros EPAs dentro de la EPN.

En definitiva, CEP agrupa las técnicas y herramientas necesarias para analizar y controlar las secuencias complejas de eventos interrelacionados que fluyen por los SI. Por tanto, con el fin de procesar y generar información útil a partir de los datos de los sensores del vehículo, la solución de diseñar una infraestructura CEP que se encargue de dicha tarea parece resultar viable.

### 6.4.2 Tecnología de Procesamiento de Eventos Complejos en el MGAI

Así, adaptando la tecnología CEP al ámbito del presente proyecto, se pueden ver los diferentes datos sensorizados procedentes del FS como los eventos del SI. Por lo tanto, el MGAI contendrá una EPN que recibirá eventos del FS y tratará de inferir los tramos de los que consta un itinerario al igual que la composición del habitáculo del vehículo en cada tramo.

Para conseguir inferir el habitáculo a través de la metodología CEP se ha construido una EPN siguiendo un modelo de Jerarquía de Abstracción de Eventos (JAE, Events Abstraction Hierarchy). Una JAE se define como una EPN formada por un conjunto de EPAs repartidos en diferentes niveles, en donde cada nivel reconoce una serie de eventos y genera eventos más complejos para ser procesados por los EPAs de los niveles superiores.

Como se puede ver en la Figura 9, que muestra la estructura de la JAE diseñada, se han creado tantos niveles de EPAs como niveles de especificidad existe en la taxonomía de ocupación descrita en la sección 6.2.3. Cada nivel será experto en una

serie de eventos e inicialmente será capaz de inferir una ocupación con un nivel de especificidad equivalente a su nivel en la JAE. De esta forma, por ejemplo, un EPA de nivel 3 en principio es capaz de determinar, en función de sus EPRs, si el ocupante en un determinado departamento del vehículo es un adulto o un menor.

Esta distinción de los EPAs se ha realizado así ya que en principio, para averiguar si un departamento está vació u ocupado simplemente con la información de las puertas es suficiente. Sin embargo para discernir si esa ocupación es una carga o una persona se necesitan más eventos (más información). Cada uno de los EPAs está especializado en un departamento del habitáculo.

Se ha buscado la coincidencia entre los distintos niveles de especificidad de la taxonomía de ocupación y los niveles de la JAE de modo que cada EPA atienda a unos eventos concretos con el fin de determinar la ocupación en función del nivel en el que se encuentre. De esta manera, conforme los eventos vayan subiendo por los niveles, las EPRs de más alto nivel serán capaces de inferir conceptos de más elevada especificidad. Los EPAs de mayor nivel deberán atender a más eventos y tendrán mayor complejidad para detectar la ocupación.

Cada vez que el patrón de un EPR se cumple, como acción asociada se genera un nuevo evento que es mandado hacía el nivel superior. Estos *nuevos eventos* se nombran como *Indicios*, y tienen una estructura de nombres bien definida tal como se ve en la Figura 10.

Por la propia característica de a JAE, cuando un EPR de un EPA en un determinado nivel se activa, es porque previamente otros EPR de EPAs en niveles inferiores se han activado generando el evento indicio que ha causado que se cumpla el patrón de la EPR. De esta manera, se pueden ver las EPRs y los EPAs de los niveles inferiores como las *causas* del nuevo evento indicio a generar.

Por tanto, cada evento indicio generado por los EPAs de la JAE a parte de un valor COcupacion, contiene como atributo una lista de las EPRs que han provocado que dicho evento se genere.

En la Figura 11 se puede ver un evento IndicioAdultoConductor. Se observa que el primera EPR que se ha activado en la JAE es el EPR 0 del EPA Puerta Conductor. Esta regla ha escuchado que la puerta del conductor se abría y cerraba y como acción asociada ha generado un IndicioOcupadoConductor. Éste evento es escuchado por el EPR1 del EPA Cint/Vent Conductor a la vez que escuchaba que se anclaba el cinturón

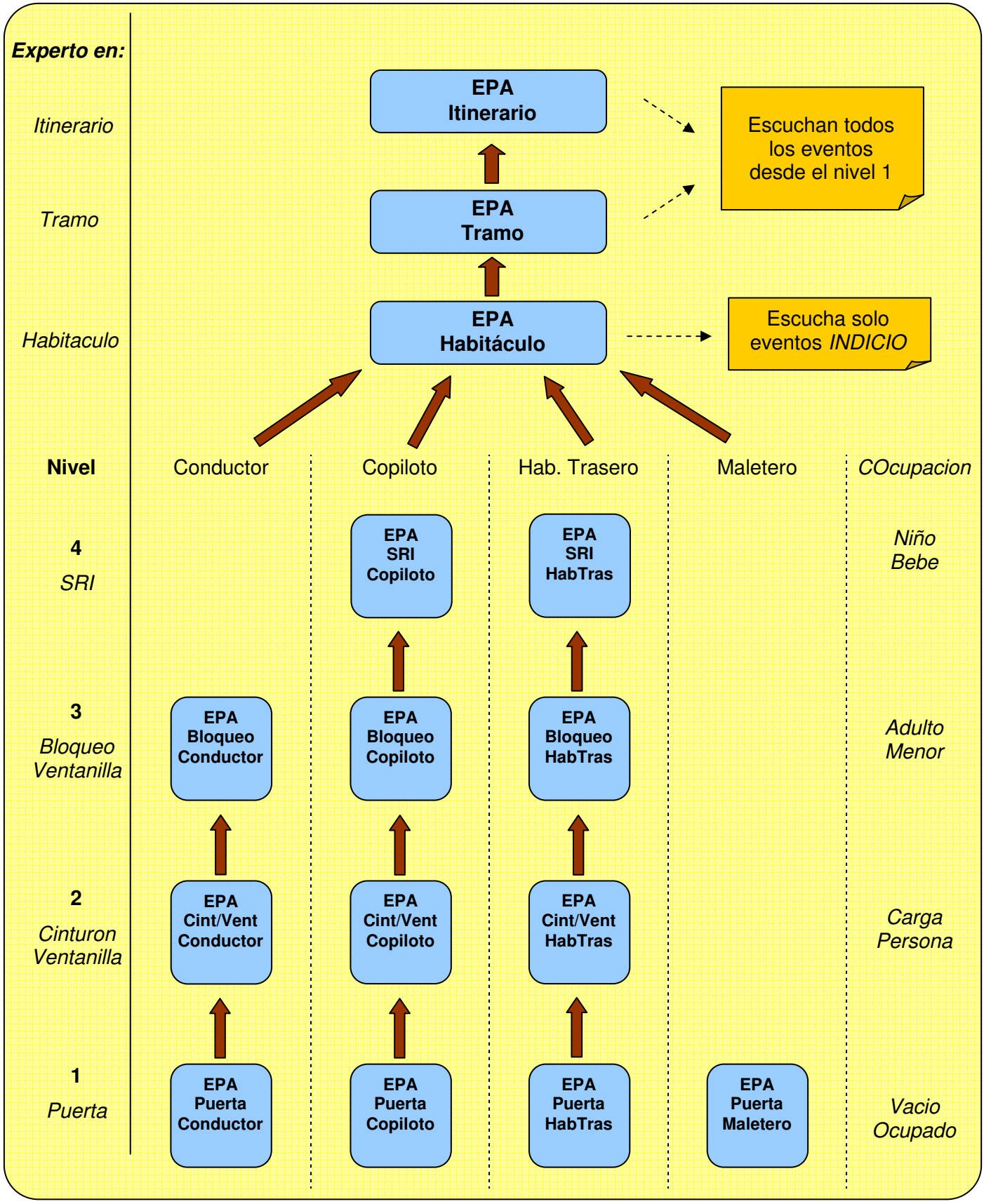



Figura 9. JAE para la inferencia de tramos, ocupaciones e itinerarios

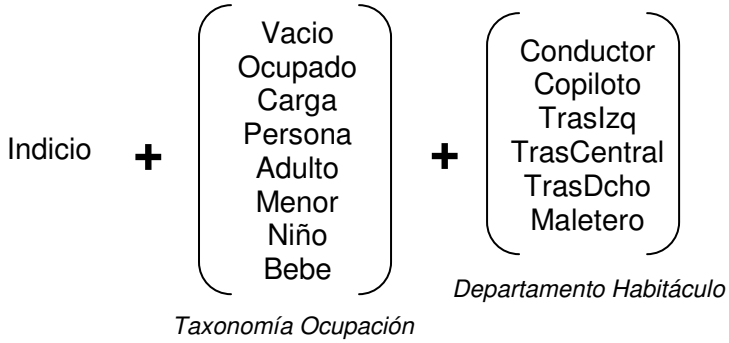



Figura 10. Estructura de nombres de los Indicios

**IndicioAdultoConductor**

*COcupacion*: Adulto
*Causas*:
- ✓ EPA Puerta Conductor – EPR 0
- ✓ EPA Cint/Vent Conductor – EPR 1
- ✓ EPA Bloqueo Conductor – EPR 0

Figura 11. Ejemplo de un evento Indicio

del conductor, con lo que genera un nuevo evento IndicioPersonaConductor. El nivel 3 ha escuchado éste último indicio y no ha tenido ninguna otra percepción que le indicase que podría haber un menor, con lo cual ha generado el IndicioAdultoConductor. Se ha podido hacer una reconstrucción completa de las distintas reglas que se han ido activando en el sistema hasta llegar al último indicio.

Teniendo ya una visión general de la JAE diseñada, se pasa a describir brevemente cada uno de sus niveles.

**Nivel 1 – Puertas**: Los EPAs a este nivel están encargados de escuchar los eventos de puertas y cerraduras que llegan a la JAE. Estos son los primeros en inferir cuántos pasajeros hay y dónde pueden estar situados. Crean eventos *IndicioVacio* e *IndicioOcupado* por cada departamento según el número de veces que escuche que sus puertas se abren y se cierran.

Existe en este nivel una problemática asociada al departamento trasero central (TC), ya que los EPAs de este nivel no son capaces de percibir si dicho departamento esta ocupado o no. Sin embargo, dicha ocupación se podrá detectar en los niveles superiores.

**Nivel 2 – Cinturones y ventanillas**: Este segundo nivel escucha eventos relacionados con los cinturones y ventanillas de cada departamento del vehículo, además de los eventos generados en el nivel 1. Si los EPAs de nivel 1 decían *cuanta* ocupación podría haber y dónde estaba situada, estos de nivel 2, afinan un poco más el cuántos y el dónde. Así, cuando escuchen un anclaje de cinturón supondrá una persona, al igual que al detectar que una ventanilla se mueve. Si por el contrario un EPA escucha un *IdicioOcupado* del nivel 1, pero no detecta posteriormente ningún síntoma de persona (anclaje de cinturón o movimiento de ventanilla) supondrá que en ese departamento hay una carga (ej, maletín, bolso,…).

**Nivel 3 – Bloqueo ventanilla**: Este nivel se encarga de escuchar los eventos *IndicioPersona* provenientes del nivel 2. Si los EPAs de este nivel no detectan ningún evento más, por defecto infieren que el habitáculo está ocupado por un adulto, generando en consecuencia el evento *IndicioAdulto*. En caso de que se haya tocado alguna ventanilla y posteriormente se detecte un bloqueo de las mismas, se inferirá que existe un menor en el departamento en el que se ha detectado movimiento de ventanilla posteriormente bloqueada, por lo que se construirá un evento *IndicioMenor* para el departamento afectado.

**Nivel 4 – SRI (Sistema de Retención Infantil)**: Este es el nivel encargado de generar las ocupaciones de mayor grado de especificidad. Es decir, una vez que se ha detectado que existe un menor en un departamento del vehículo, distinguir si es un niño o un bebe. Por defecto, los EPAs de este nivel, al recibir un evento *IndicioMenor* en este nivel supondrán que es un Niño, aunque si por el contrario, detecta una sillita de retención infantil en su departamento, inferirá que la ocupación es un bebe. Los EPAs suponen que en su departamento hay una silla de retención infantil, cuando detectan que hay un anclaje de cinturón activo antes de que se haya abierto cualquiera de las cerraduras del vehículo por primera vez. Este nivel crea, como se ha visto, *IndicioNiño* e *IndicioBebe*.

Se ha dicho hasta ahora que los eventos indicios que genera cada EPA van en función del nivel en el que se encuentran, aunque realmente este comportamiento no es del todo cierto. Cuando se activa una EPR de un EPA, el evento indicio que genera lo decide el MMUC. El comportamiento descrito hasta el momento va a ser el comportamiento por defecto del sistema, ya que el perfil inicial de usuario se va a configurar de tal modo que los EPAs generen eventos indicio equivalentes a su nivel tal y como se ha descrito en esta sección. No obstante, el sistema se ha de adaptar a los hábitos de cada usuario y puede darse el caso de que para un usuario concreto

pueda inferirse ocupaciones de nivel de especificidad 4 desde un EPA de nivel 1, siendo la información del nivel 1 suficiente una vez conocidos los hábitos del usuario. Todo este proceso de describe en profundidad en la sección 6.5.2.

**Nivel Habitáculo**: Al contrario que el resto de niveles, este nivel está formado por un solo EPA encargado de escuchar los eventos *Indicio* generados por el resto de EPAs de la JAE. Su cometido es el de rellenar el habitáculo del coche que se ha inferido hasta ese momento a partir de los eventos *Indicio* generados por los EPAs. Para ello, cuando escucha un *Indicio*, busca el departamento del coche al que se refiere y lo coloca en su posición, machacando el que ya hubiera. Esto da como resultado una estructura con los distintos departamentos del coche, y en cada uno habrá un evento *Indicio* conteniendo la ocupación que se ha inferido para él.

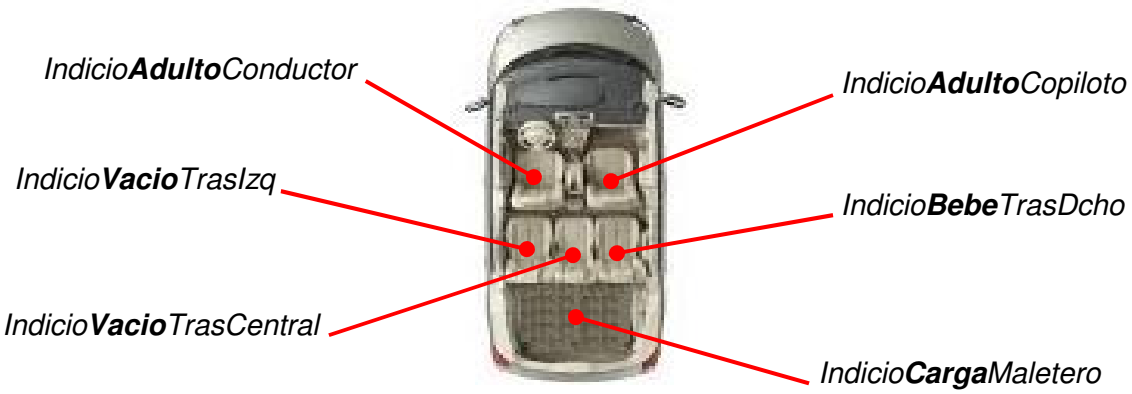



Figura 12. Ejemplo de habitáculo inferido

**Nivel Tramo:** Al igual que en el caso anterior, este nivel consta de un único EPA encargado de detectar el final de un tramo dentro del itinerario. Para considerarse que se ha llegado al final de un tramo busca que ocurran cualquiera de estas las siguientes situaciones:

- Que una vez que el coche ha estado en marcha (velocidad distinta de 0), se haya parado (velocidad igual a 0) y se abra cualquiera de las puertas del coche.
- Que una vez que el coche ha estado en marcha (velocidad distinta de 0), se haya parado (velocidad igual a 0) y esté a ralentí más de dos minutos.
- Que una vez que el coche ha estado en marcha (velocidad distinta de 0), se haya parado (velocidad igual a 0) y se apague el motor.

Una vez que se ha detectado el final de tramo se lanzará el MMUC para mostrar al usuario el habitáculo inferido para el tramo que se acaba de detectar.

**Nivel Itinerario**: Este EPA detecta el final de Itinerario. Es el encargado de lanzar el proceso de finalización de la ejecución, el cual genera un archivo XML para almacenar el Itinerario inferido con todos los tramos y eventos por tramos.

Quizá los patrones y reglas que se están intentando detectar aquí no son tan complejos como para requerir tecnología CEP. Sin embargo no hay que olvidar que esto forma parte de un trabajo más ambicioso donde en un futuro se tratarán más fuentes de información externas. Además, siempre se pueden incluir patrones más complicados si se percibe que los actuales no funcionan.

## *6.5 MMUC*

A la hora de utilizar un vehículo es fácil imaginar que cada usuario lo hará con unos hábitos de comportamiento totalmente diferentes a los del resto. Además, para un determinado usuario, dichos hábitos pueden variar en función de determinadas circunstancias externas. Así, por ejemplo, una determinada persona puede estar acostumbrada a dejar en el asiento trasero izquierdo su maletín cada vez que va al trabajo, mientras otro individuo viaja con su hijo pequeño en ese mismo asiento todas las mañanas para dejarlo en el colegio. Sin embargo, lo único que los sensores del coche detectan es que ha abierto y cerrado la puerta trasera izquierda. Por lo tanto, un sistema rígido que asumiera que la actividad abrir y cerrar dicha puerta siempre implica que hay una persona en el asiento trasero izquierdo no es viable. Por el contrario, el sistema debe ser capaz de adaptarse a los hábitos de los diferentes usuarios y determinar un perfil de comportamiento para cada uno de ellos. En otras palabras, debe ser capaz de adaptar el comportamiento de cada usuario a partir de sus interacciones con la carrocería. Esto provocará unas inferencias mucho más precisas por parte del sistema respecto al tipo de ocupación de cada departamento en el habitáculo.

Por todo ello, se ha decidido diseñar una primera aproximación de un mecanismo adaptativo que permite al sistema modelar el comportamiento de los diferentes usuarios con su vehículo. Para ello, el sistema se vale de las validaciones y correcciones que los usuarios realizan sobre las inferencias realizadas por el MGAI. Este mecanismo de adaptación y modelado de usuario se ha realizado inspirándose en el enfoque del aprendizaje por refuerzo.

### 6.5.1 Introducción al aprendizaje por refuerzo

El aprendizaje por refuerzo se entiende como aquel aprendizaje que sigue un esquema de ensayo-error a través del cual un agente computacional aprende a realizar en cada momento el conjunto de acciones más deseables a partir de su interacción con el entorno dinámico que lo rodea [23] y [24]. Este tipo de aprendizajes suelen basarse en técnicas probabilísticas o en la programación dinámica [25] y han sido utilizados en ámbitos tan diversos como la resolución de juegos o el control de robots autónomos.

#### 6.5.1.1 Modelo Subyacente

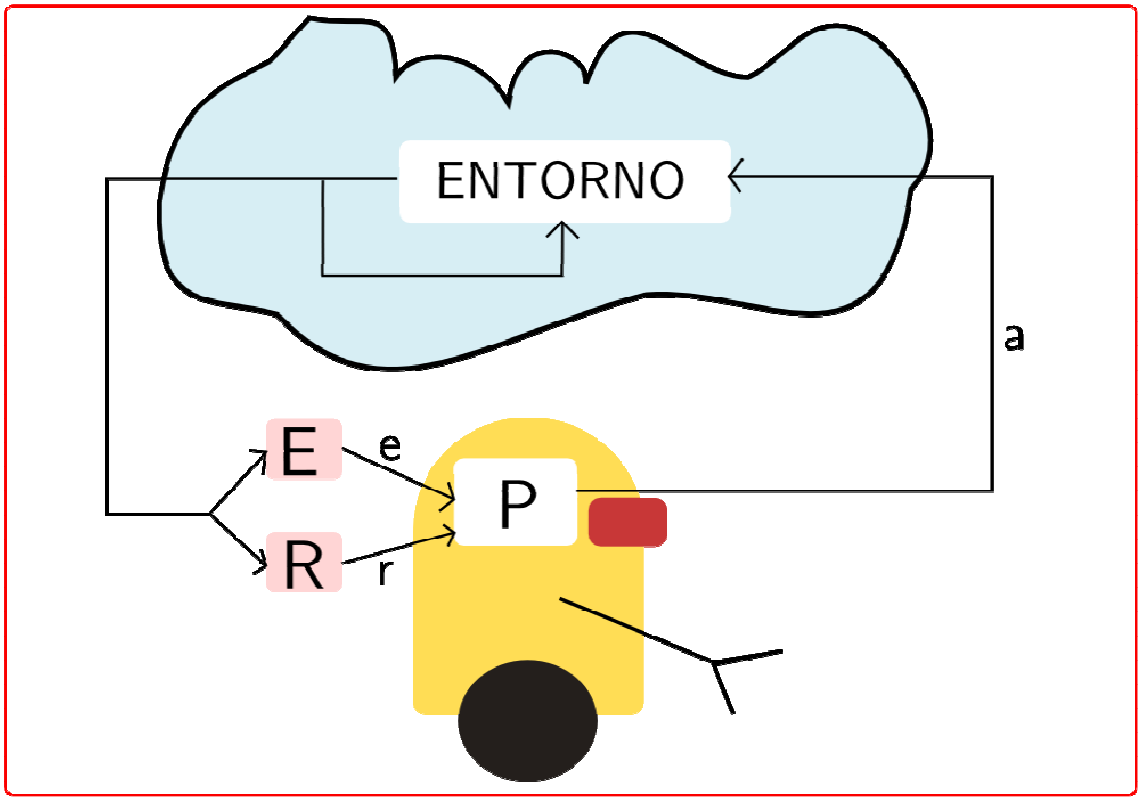



Figura 13. Modelo del aprendizaje por refuerzo

El modelo que subyace en el aprendizaje por refuerzo se muestra en la Figura 13. En él, un agente opera en un determinado Entorno y recibe como entrada el estado actual del entorno (e) y, en función de su política (P), la cual realiza un mapeo entre el estado del entorno y la acción más apropiada a realizar, genera una acción (a) como salida con el objetivo de aumentar una función valor. Esa acción modifica el estado del entorno lo que a su vez genera una señal de refuerzo (r) hacia el agente. De esta manera, la política del agente debe intentar generar las acciones que permitan hacer lo más grande posible la suma de las señales de refuerzo que recibe.

Este modelo ha servido como base para diseñar una primera aproximación de un mecanismo de adaptación. Pese a todo, a la hora de trasladar el aprendizaje por refuerzo al ámbito del presente trabajo, se encuentran algunas similitudes con respecto al modelo original pero también con importantes diferencias tal y como se discute en la sección 6.5.2.

### 6.5.2 Adaptación del aprendizaje por refuerzo al problema planteado

Tal y como se ha visto en la sección 6.5.1.1, el aprendizaje por refuerzo se basa en el hecho de que un agente que interacciona con un entorno dinámico que le rodea. Esto es fácilmente extrapolable al ámbito de la interacción usuario-carrocería, en donde ese agente podría tratarse de nuestro MGAI, mientras que el entorno dinámico consistiría en las diferentes actividades (transformadas en eventos) que los usuarios realizan dentro del vehículo.

Sin embargo, entre el enfoque del presente proyecto y el aprendizaje por refuerzo existen dos importantes diferencias:

- Mientras que en el aprendizaje por refuerzo el agente intenta modificar el entorno que le rodea mediante sus acciones, en el presente proyecto las acciones del mecanismo de adaptación intentan inferir el habitáculo del vehículo de la manera más precisa posible sin intención alguna de modificar su entorno, el cual depende única y exclusivamente de las diferentes actividades que el usuario realiza dentro del vehículo.

- Las señales de refuerzo que recibirá el agente del sistema desarrollado no vendrán dadas por el entorno, sino por el usuario al validar y corregir el habitáculo inferido, por lo que de esta manera el sistema desarrollado se aproxima a los sistemas de aprendizaje supervisados.

Otro aspecto clave a tener en cuenta es que esta primera aproximación del mecanismo de adaptación deberá distinguir entre los diferentes usuarios, para poder realizar una adaptación personalizada a cada uno de ellos y afinar el comportamiento del sistema en función de sus características.

Por último, en el presente trabajo, la adaptación para cada usuario no distingue entre los diferentes itinerarios que éste puede efectuar. Para poder realizar dicha distinción deberían incorporarse nuevas variables al sistema que permitieran

establecer el itinerario concreto en el que el usuario se encuentra en cada momento, lo cual se deja como trabajo futuro tal y como se discute en la sección 9.

Asumiendo todas estas restricciones, el diseño de la primera versión del mecanismo adaptativo se discute en la siguiente sección.

#### 6.5.2.1 Diseño del mecanismo de adaptación

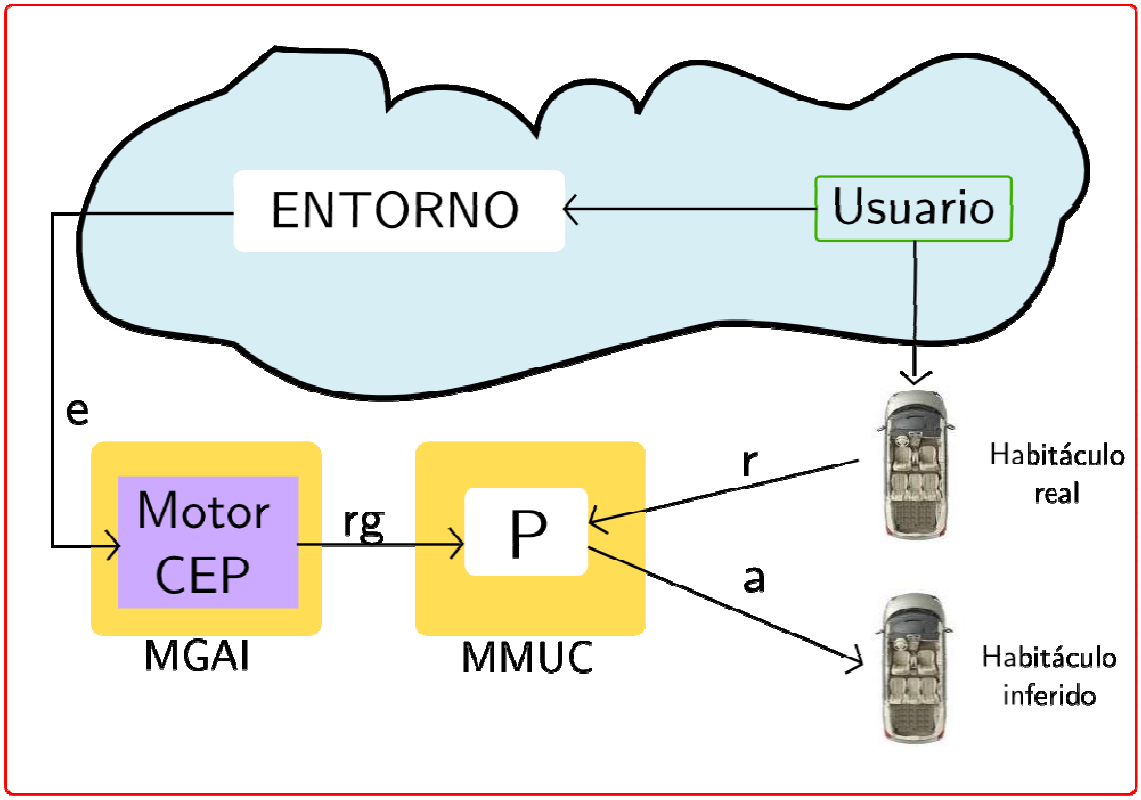



Figura 14. Diseño simplificado mecanismo de adaptación

En la Figura 14 se muestra el diseño general de la primera versión del mecanismo de adaptación. En ella se puede ver que el estado del entorno (e) es procesado por el MGAI la cual, indica al mecanismo de adaptación qué EPR de qué EPA ha saltado (rg) en función de estado del entorno. A continuación la política (P) del mecanismo de adaptación obtiene la acción a realizar (a) que modificará el habitáculo inferido. Por último, el usuario mediante la especificación de la configuración del habitáculo real al final de cada tramo (ver sección 7.5), generará la señal de recompensa (r). Se pasa a describir más detalladamente cada elemento de la Figura 14.

1) Estado del entorno (e) y regla activada (rg)

La información de estado del entorno (e) consistirá en los diferentes eventos que se van generando por las actividades del usuario dentro del coche durante cada tramo

(ver sección 6.1.4). Dichos eventos son procesados por el MGAI que, en caso de que se activara alguna de sus reglas, informaría al mecanismo de adaptación qué regla de qué EPA se ha activado (rg). Es este sentido hay que aclarar que cada regla lleva asociada uno de los departamentos del habitáculo del vehículo.

2) Señal de refuerzo (r)

Cada vez que el usuario corrige los posibles errores que el sistema ha cometido en el habitáculo inferido, obteniéndose así el habitáculo real, el mecanismo de adaptación recibe como señales de refuerzo los conceptos ocupación de la taxonomía de ocupaciones (en adelante COcupacion) que representan el tipo de ocupación real que hay en cada departamento del habitáculo. Todos estos COcupacion son utilizados para afinar la política del sistema.

3) Política del mecanismo de adaptación (P) y acción inferida (a)

Tal y como se comenta en la sección 6.5.1.1, la política de un agente realiza un mapeo entre su entrada y la acción (a) más apropiada para ella. En este caso, la entrada del sistema es la regla que ha saltado en el motor CEP, mientras que la acción será el COcupacion encargado de rellenar un determinado departamento del habitáculo inferido. Por tanto la política del sistema establece qué tipo de ocupación debe de inferirse para el departamento del habitáculo asociado a la regla recibida como entrada.

Por otro lado, dicha política se ve refinada por los COcupacion provenientes del habitáculo real que hacen el papel de señales de refuerzo. Para ello se hace uso de una heurística probabilista que permite ajustar las inferencias del sistema.

Teniendo en cuenta que la política es el eje central del mecanismo de adaptación, es explicada más detalladamente en la siguiente sección.

#### 6.5.2.2 Funcionamiento y estructura de la política del mecanismo de adaptación

A la hora de presentar la política de adaptación diseñada, se pueden distinguir tres aspectos fundamentales de la misma. Por un lado el cómo se realiza el mapeo regla-COcupacion, por otro el método por el cual se elije un COcupacion determinado como acción de salida, y finalmente la heurística que permite ajustar el comportamiento de la política en función de las señales de refuerzo recibidas. Estos tres puntos son desarrollados a continuación.

1) Mapeo regla-COcupacion

Cada una de las reglas de todos los EPAs contenidos en el motor CEP debe tener asociada, dentro del mecanismo de adaptación, un conjunto de posibles COcupacion que puedan se seleccionados como acción de salida. Con el objetivo de hacer el sistema lo más flexible y dinámico posible, se asume que una regla puede generar como acción cualquier tipo de COcupacion de cualquier nivel de especificidad independientemente de en qué nivel se encuentre el EPA al que pertenezca. Así, el mecanismo de adaptación, al recibir como entrada la activación de una determinada regla de un EPA de nivel 1, podría generar como salida un COcupacion de nivel de especificidad 4. Esta decisión reporta dos importantes beneficios:

- Permite al sistema inferir un COcupacion con un nivel de especificidad mayor al inicialmente asignado a la regla que se toma como entrada. En caso de no hacerse así, si por ejemplo un usuario sólo realizara las actividades de abrir y cerrar puertas, recogidas por los EPAs de nivel de 1, el sistema solo sería capaz de inferir que los diferentes departamentos del habitáculo se encuentran ocupadas o vacías, sin poder entrar a un mayor nivel de especificidad, ya que los EPAs de ese nivel tiene asignado por defecto los COcupacion vacío u ocupado. Sin embargo, al permitir que cualquier regla pueda inferir cualquier tipo de COcupacion, el sistema podrá realizar inferencias de un grano más fino.

- El sistema puede ajustar el nivel de especificidad de sus inferencias a un cierto grado de detalle que quiera el usuario. Así, si a través del habitáculo real el usuario nunca especifica COcupaciones por encima de un determinado nivel de especificidad, el sistema comprende que el usuario solo está interesado en inferencias hasta ese nivel concreto, por lo que lo tendrá en cuenta en futuras inferencias. Este punto puede verse como la base para, en futuras versiones de la aplicación, implementar un sistema de control de la privacidad que permita al usuario especificar hasta que niveles de especificidad quiera que el sistema llegue.

Con todo ello, cada EPR de los EPAs, lleva asociada en el mecanismo de adaptación una estructura de datos llamada *jerarquía de probabilidades* (JP) tal y como se muestra en la Figura 15. Cada JP de cada EPR está compuesta por los siguientes campos:

- Niveles de Especificidad (NE): Indica los diferentes niveles de especificidad que pueden tener los consecuentes de una regla. Cada NE contiene a su vez los

campos:

- Hijo Izquierdo (HI): *COcupacion* que representa al nodo izquierdo en el nivel de especificidad indicado (por ejemplo el concepto *Carga* en el nivel de especificidad 2, o el concepto *Adulto* en el nivel de especificidad 3).

- Hijo Derecho (HD): *COcupacion* que representa al nodo derecho en el nivel de especificidad indicado (por ejemplo el concepto *Persona* en el nivel de especificidad 2, o el concepto *Menor* en el nivel de especificidad 3).

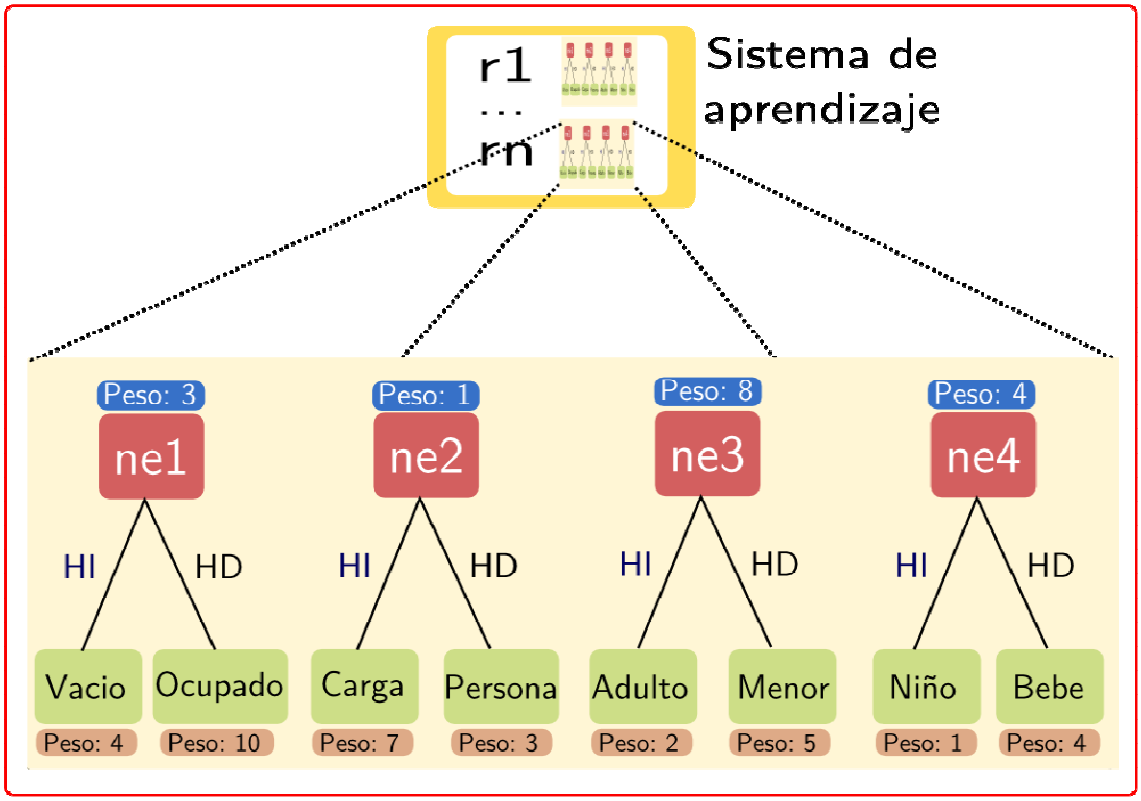



Figura 15. Jerarquía de probabilidades asociada a cada regla

En la Figura 15 también se aprecia que tanto los NE como los HD e HI llevan asociados unos *pesos*. Dichos valores son utilizados por el mecanismo de selección con el fin de generar el *COcupacion* que constituye la acción de salida. Teniendo en cuenta que el valor de dichos pesos estarán personalizados para cada usuario, la JP de cada regla deberá estar replicada para cada uno de los usuarios que hagan uso del sistema.

2) Mecanismo de selección

El algoritmo de selección utilizado para elegir qué *COcupacion* concreto utilizar como acción sigue un enfoque probabilístico. En él, los pesos asignados a los HD y HI dentro de la JP establecen cuánto es de probable que el *COcupacion* que representan sea el elegido, de tal manera que a más peso, más probabilidad de salir elegido.

Cuando la JP se inicializa, a sus diferentes elementos le son asignados unos pesos por defecto. Para dicha inicialización, se tiene en cuenta el tipo de regla asociada a la JP, distribuyendo los pesos mediante un conocimiento intuitivo.

En el momento que el mecanismo de adaptación recibe como entrada una regla que acaba de activarse en uno de los EPAs, éste accede a la JP asociada a la regla entrante. A continuación genera un número aleatorio para cada par de hijos dentro de cada nivel de especificidad. En función de dicho número aleatorio y de los pesos asociados a cada uno de los hijos, se selecciona un *hijo candidato* (HC). Posteriormente, se genera otro nuevo número aleatorio, independiente del anterior, que permite seleccionar qué nivel de especificidad concreto utilizar. Para ello se tienen en cuenta los pesos asociados a cada nivel de especificidad. Finalmente, se utiliza como acción el *COcupacion* del HC del nivel de especificidad seleccionado.

El pseudocódigo del algoritmo es el siguiente:

**Entrada:**

- $JP_k^i$: Jerarquía de probabilidades asociada a la regla $r^i_{EPA_k^j}$ ($i \in \{1..n\}$, $j \in \{1..4\}$, $k \in \{C,CP,TI,TC,TD,M\}$) recibida como entrada por el sistema.

**Salida:**

- $COcupacion^k_{\inf}$, $k \in \{C,CP,TI,TC,TD,M\}$: *COcupacion* generado como acción para modificar el habitáculo inferido.

**Funciones Auxiliares:**

- `SeleccionarHijo(NE ne)`: Función que selecciona a uno de los dos hijos del nivel de especificidad `ne` teniendo en cuenta el peso de cada uno de ellos, en el que a más peso mayor probabilidad de salir elegido.

▪ `SeleccionarEspecificidad(JP jp)`: Función que selecciona uno de los niveles de especificidad de la jerarquía de probabilidades `jp` en donde la especificidad con más peso tiene mayores opciones de salir elegida.

**Pasos del algoritmo:**

1. `Desde` $index$ = `1 hasta 4 hacer`

   $JP_k^i . NE[\ index\ ].HC$ = `SeleccionarHijo`$(JP_k^i . NE[\ index\ ])$

2. $index$ = `SeleccionarEspecificidad`$(JP_k^i)$

3. $COcupacion_{\inf}^k = JP_k^i . NE[\ index\ ].HC$

4. `return` $COcupacion_{\inf}^k$

Este método guiado por los pesos de los diferentes elementos dentro de la JP tiene dos grandes ventajas:

- Por un lado, al realizarse la elección en base a un número aleatorio, se evita que el método de selección se estanque en algún máximo local. De esta forma, cualquier elemento de una JP tiene la posibilidad de poder ser seleccionado como acción de salida.

- En segundo lugar, al ser un método sencillo, no es computacionalmente costoso, lo cual es deseable teniendo en cuenta que el sistema operará como un equipamiento on-board dentro del vehículo.

3) Afinamiento de la política del sistema

Tal y como se ha visto en la sección anterior, un elemento crucial para seleccionar un *COcupacion* es el peso que tanto él como su NE tengan en la JP. Por tanto, una manera de afinar la política de selección del sistema consiste en modificar los pesos de los diferentes elementos de la JP de tal manera que se otorgue más peso a aquellas *COcupaciones* que el usuario indica repetidamente, a través de sus correcciones, que son las que realmente existen en cada departamento del habitáculo.

De esta forma, los pesos dentro de la JP se van modificando en función de los errores y los aciertos de las inferencias realizadas. En este sentido hay que señalar que debido a que los *COcupacion* constituyen una taxonomía, hay que tener en cuenta en qué posiciones dentro de la misma se encuentran el *COcupacion* inferido y aquél indicado por el usuario como la verdadera ocupación, ya que, aunque ambos fueran diferentes, constituye un error mucho menos grave si ambos están en la misma rama

de la taxonomía. Así, por ejemplo, el que usuario indicara que en un determinado departamento hay un bebé, cuando el sistema ha inferido que estaba vacío es mucho más grave que si el sistema hubiera inferido que había una persona, ya que un bebé es también una persona. Además, cuando más cerca esté la inferencia al nivel de especificidad del *COcupacion* indicado por el usuario, mejor. Así, por ejemplo es más deseable haber inferido menor a haber inferido persona en el caso de que la ocupación real fuera niño.

Teniendo en cuenta lo anterior, debemos tener claro en qué situaciones se debe *premiar* (aumentar su peso) o *penalizar* (disminuirlo) el *COcupacion* inferido por el usuario. En este sentido, se puede distinguir cuatro situaciones diferentes derivadas de lo inferido por el sistema ($COcupacion_{\inf}$), lo indicado por el usuario ($COcupacion_u$) y sus niveles de especificidad ($COcupacion_{\inf}.ne$, $COcupacion_u.ne$) para cada departamento del habitáculo:

A) $COcupacion_{\inf} = COcupacion_u$: En el caso que lo inferido y lo indicado por el usuario coincidan totalmente, debemos recompensar tanto al nivel de especificidad del $COcupacion_{\inf}$ como al propio $COcupacion_{\inf}$ incrementando el peso de ambos. De esta forma, ambos ganan en posibilidades de volver a salir elegidos como salida del sistema en el futuro. Además, se premia a los HD de todos los niveles de especificidad por debajo de $COcupacion_{\inf}.ne$. Con esto facilitamos que, si en futuras ejecuciones el NE seleccionado es diferente a $COcupacion_{\inf}.ne$, al menos sea más probable elegir un *COcupacion* en la misma rama que $COcupacion_{\inf}$.

B) $COcupacion_{\inf} \neq COcupacion_u$ y $COcupacion_{\inf}.ne = COcupacion_u.ne$: Esta situación se presenta cuando el sistema ha escogido correctamente el nivel de especificación pero se ha equivocado de hijo dentro del mismo. Por ejemplo, cuando la salida del sistema ha sido adulto cuando la real era menor. Por lo tanto, el sistema debe penalizar al $COcupacion_{\inf}$ pero también a su nivel de especificidad, ya que las inferencias a dicho nivel han demostrado no ser todavía *fiables*. Al igual que en el caso anterior, se premian a los HD de los NE por debajo de $COcupacion_u.ne$. Además, se comprueban los HC de esos mismos NE, premiando al NE mas cercano a $COcupacion_u.ne$ cuyo HC se encuentre en la rama camino de $COcupacion_u$. De esta forma se favorece que

el sistema genere como salidas *COcupaciones* a niveles de especificidad más bajos dentro de la taxonomía. En otras palabras, si a un NE 3 el sistema no resulta fiable a la hora de decidir si la ocupación es adulto o menor, pero en el nivel 2 el HC ha sido persona, se penaliza el NE 3 y se premia al NE 2 y a su HC para intentar que en las siguientes ejecuciones la salida sea simplemente persona.

C) $COcupacion_{\inf} \neq COcupacion_u$ y $COcupacion_{\inf}.ne > COcupacion_u.ne$: En este caso, la inferencia del sistema ha ido *más allá* a lo indicado por el usuario, al realizar una inferencia con un nivel de especificidad superior a lo indicado por el usuario. Esto sucede, por ejemplo, cuando $COcupacion_u$ es ocupado y $COcupacion_{\inf}$ es bebé. Dicha situación no es deseable, ya que el sistema debe intentar adaptarse a los niveles de especificidad que el usuario le va indicando. Por tanto, la primera acción a realizar en este caso debe de ser la penalización de todos los niveles de inferencia por encima de $COcupacion_u.ne$. Así, se intenta evitar que en futuras ejecuciones vuelva a cometerse el mismo tipo de error. A continuación, se premian todos los HD de los NE por debajo de $COcupacion_u.ne$ o lo que es lo mismo, todos los *COcupaciones* que existen en el camino desde la raíz de la taxonomía hasta el tipo de ocupación indicado por el usuario. Por último, y al igual que en C) se comprueban los HC de los NE inferiores a $COcupacion_u.ne$ y se premia a aquel NE con nivel más alto cuyo HC se encuentre en la misma rama que $COcupacion_u$.

D) $COcupacion_{\inf} \neq COcupacion_u$ y $COcupacion_{\inf}.ne < COcupacion_u.ne$: En esta última situación, el sistema se ha *quedado corto* ya que el usuario ha indicado un tipo de ocupación en el departamento con un NE mayor al indicado por el sistema. Un ejemplo de este hecho ocurre cuando $COcupacion_{\inf}$ es carga y $COcupacion_u$ es bebé. Las acciones para premiar o penalizar los distintos elementos dentro de la JP en este caso son muy parecidas a las ya comentadas anteriormente. Por un lado, premiar los HD de los NE por debajo de $COcupacion_u.ne$, por otro recompensar al NE cuyo HC esté en la misma rama que $COcupacion_u$ y finalmente penalizar a los NE que se encuentren por encima del NE de $COcupacion_u$.

A partir de las diferentes acciones a realizar en los cuatro casos detallados anteriormente, el pseudocódigo del algoritmo para el afinamiento de la política queda como sigue:

**Entrada:**

- $COcupacion_u^k$, $k \in \{C,CP,TI,TC,TD,M\}$: Conceptos ocupación proporcionados por el usuario con la ocupación *real* de cada departamento del habitáculo.

- $COcupacion_{\inf}^{ik}$, $i \in \{1..n\}$, $k \in \{C,CP,TI,TC,TD,M\}$: Conceptos ocupación inferidos por el motor CEP para cada departamento del habitáculo.

**Salida:**

- $JP_k^i$, $i \in \{1..n\}$, $k \in \{C,CP,TI,TC,TD,M\}$: JPs de las reglas del motor CEP ajustadas a los fallos o aciertos detectados.

**Funciones Auxiliares:**

- `PremiarNodo(`*`COcupacion`* `co)`: Función que premia al *`COcupacion`* `co` en una JP.

- `PremiarEspecificidad(`*`NE`* `n)`: Función que premia al *`NE`* `n` de la JP de una determinada regla.

- `ObtenerRama(`*`NE`* `n,` *`COcupacion`* `co)`: Función que devuelve cuál de los dos hijos del *`NE`* `n` (HD o HI) se encuentra en la rama que va camino del *`COcupacion`* `co`. Así por ejemplo, la llamada `ObtenerRama(1, Bebe)` devolvería *`Ocupado`*, o la llamada `ObtenerRama(2, Niño)` devolvería *`Persona`*.

- `CastigarNodo(`*`COcupacion`* `co)`: Función que castiga al *`COcupacion`* `co` en una JP.

- `CastigarEspecificidad(`*`NE`* `n)`: Función que castiga al *`NE`* `n` de la JP de una determinada regla.

- `EsRama(`*`COcupacion`* `co1,` *`COcupacion`* `co2)`: Función que determina si `co1` se encuentra en la rama que va camino de `co2`. Así, la llamada `EsRama(Persona, Adulto)` devolvería `true`, mientra que `EsRama(Vacio, Niño)` devolverá `false`.

**Pasos del algoritmo:**

Para cada $k \in \{C,CP,TI,TC,TD,M\}$ hacer // Se recorren los habitáculos uno a uno

1. flag1 = flag2 = false
2. Si $COcupacion_u^k.ne == COcupacion_{\inf}^{ik}.ne$ entonces

   // Caso A

   2.1 Si $COcupacion_u^k == COcupacion_{\inf}^{ik}$ entonces

      2.1.1 PremiarNodo($COcupacion_{\inf}^{ik}$)

      2.1.2 PremiarEspecificidad($COcupacion_{\inf}^{ik}.ne$)

      2.1.3 Desde $l = COcupacion_{\inf}^{ik}.ne - 1$ hasta 1 hacer

         PremiarNodo(ObtenerRama($JP_k^i.NE[l]$, $COcupacion_u^k$))

   // Caso B

   2.2 Si $COcupacion_u^k \neq COcupacion_{\inf}^{ik}$ entonces

      2.2.1. CastigarNodo($COcupacion_{\inf}^{ik}$)

      2.2.2. CastigarEspecificidad($COcupacion_{\inf}^{ik}.ne$)

      2.2.3. Desde $l = COcupacion_{\inf}^{ik}$ - 1 hasta 1 hacer

         PremiarNodo(ObtenerRama($JP_k^i.NE[l]$, $COcupacion_u^k$))

         Si EsRama($JP_k^i.NE[l].HC$, $COcupacion_u^k$) y flag1==false entonces

         PremiarEspecificidad($JP_k^i.NE[l]$)

         flag1 = true

   // Casos C y D

3. Si $COcupacion_u^k \neq COcupacion_{\inf}^{ik}.ne$ entonces

   3.1 Desde $l = COcupacion_u^k.ne + 1$ hasta 4 hacer

      3.1.1. CastigarEspecificidad($JP_k^i.NE[l]$)

   3.2 Desde $l = COcupacion_u^k.ne$ hasta 1 hacer

      3.2.1. PremiarNodo(ObtenerRama($JP_k^i.NE[l]$, $COcupacion_u^k$))

      3.2.2. Si EsRama($JP_k^i.NE[l].HC$, $COcupacion_u^k$) y flag2 == false entonces

         PremiarEspecificidad($JP_k^i.NE[l].HC$)

         flag2 = true

Como se aprecia, el algoritmo es independiente de la manera en que se premien o castiguen los diferentes elementos de una JP. En la actual versión las funciones `PremiarNodo` y `PremiarEspecificidad` simplemente incrementan en uno el valor del peso del hijo o NE que se le pase por parámetro, mientras que `CastigarNodo` y `CastigarEspecificidad` decrementan en uno el valor del paso. Sin embargo, como trabajo futuro, estas funciones pueden establecer mecanismos de recompensa y castigo más sofisticados que permitan una adaptación mucho más precisa.

### 6.5.3 Ejemplos de funcionamiento

A continuación se muestran dos ejemplos de cuál sería el funcionamiento del mecanismo de adaptación según lo explicado en los apartados anteriores frente a dos casos de uso que ilustran dos comportamientos de usuario muy diferentes el uno el otro. En el primero de ellos, un usuario, en un itinerario concreto, siempre especifica el mismo tipo de ocupación para un determinado departamento del habitáculo. En el segundo, el usuario varía el tipo de ocupación cada vez, pero siempre con el mismo nivel de especificidad.

#### 6.5.3.1 Caso de uso 1

Supóngase un usuario que, por las circunstancias que sean, las únicas actividades de carrocería que realiza a la hora de utilizar el coche son las relacionadas con abrir y cerrar las puertas. Además, cada vez que utiliza el vehículo siempre lleva un adulto como acompañante en el habitáculo del copiloto. Esta situación se repite durante un indeterminado número de veces consecutivas.

En este caso, las únicas reglas que se activarían en el sistema serían aquellas de los EPAs de nivel 1, que son los encargados de detectar los eventos generados por las actividades asociadas a abrir y cerrar las puertas. Suponiendo que hubiera una regla que se activara cada vez que la puerta del departamento del copiloto se abriera y cerrara, la Figura 16 muestra la evolución de los pesos asociados a su JP a lo largo del tiempo.

Dicha figura muestra que en la inicialización de la JP (Figura 16 (a)), el NE que tiene asociado más peso es el NE 1. Esto es así porque la regla asociada a la JP pertenece a un EPA de nivel 1, a los que por defecto se asignan los *COcupacion* con NE 1. Además, dentro de dicho NE el hijo que más peso tiene (y por tanto más posibilidades de salir elegido) es el HD (ocupado) ya que, intuitivamente, cuando se abre y cierra la puerta de un vehículo es porque se quiere dejar algo en su

departamento asociado. El resto de NEs llevan asociados pesos decrecientes conforme se alejan del NE 1.

La primera vez que se activa la regla (Figura 16 (b)), la JP ejecuta su método de selección. Así, los NC de cada NE son mostrados en color lila, mientras que el *COcupacion* que finalmente ha sido elegido como acción de salida se muestra en verde claro. En esta primera activación, la salida del mecanismo de adaptación es ocupado, lo cual es normal teniendo en cuenta que tanto su peso como el de su NE son los más altos de toda la JP.

Posteriormente, cuando el usuario indica que la ocupación real (*COcupacion*$_{U)}$ es adulto, el algoritmo de afinamiento entra en acción con el fin de modificar los pesos de la JP de manera apropiada. En este sentido los pesos modificados en cada paso son resaltados en amarillo en la figura. Teniendo en cuenta que lo indicado por el usuario (adulto) es diferente a lo inferido por el sistema (ocupado) y con un nivel de especificidad mayor (3 frente a 1), se está en el caso D de las cuatro situaciones detalladas en la sección 6.5.2.2 apartado 3). Por tanto, las acciones a realizar son, por un lado penalizar a los NE por encima de 3, lo cual queda reflejado en disminuir el peso del NE 4 de 2 a 1. Por otro, premiar a los HD desde el nivel 2 hasta el nivel 1 y al *COcupacion* adulto del NE 3 incrementando sus pesos en 1. Y por último, premiar al NE más cercano a adulto suyo HC esté en la misma rama que dicho tipo de ocupación. En este caso, el propio NE 3 eligió a adulto como HC, por lo que es el NE recompensado viendo incrementado su peso en una unidad más pasando de las 3 unidades de su inicialización a 4.

En la segunda activación de la regla (Figura 16 (c)), esta vez es persona la *COcupacion* inferida por la JP y de nuevo adulto la indicada por el usuario. Al estar en un escenario parecido al de la primera activación (ocupaciones inferidas y reales diferentes, con la real con mayor nivel de especificidad), las acciones a realizar son las mismas que en la anterior ocasión. Esto es, penalizar a los NE por encima del NE 3, lo que conlleva reducir el peso de NE 4 de 1 a 0. A continuación, premiar a los HD de los NE 2 y 1 y al *COcupacion* adulto del NE 3. Y por último comprobar el NE más alto cuyo HC esté en la rama hacia adulto. En este caso, el mecanismo seleccionó como HC para el NE 3 el *COcupacion* menor, por lo que dicho NE no es recompensado. Por contra el NE 2 tomó como HC persona que al estar en la misma rama de la taxonomía que adulto, provoca que dicho NE sea recompensando incrementándose de 4 a 5.

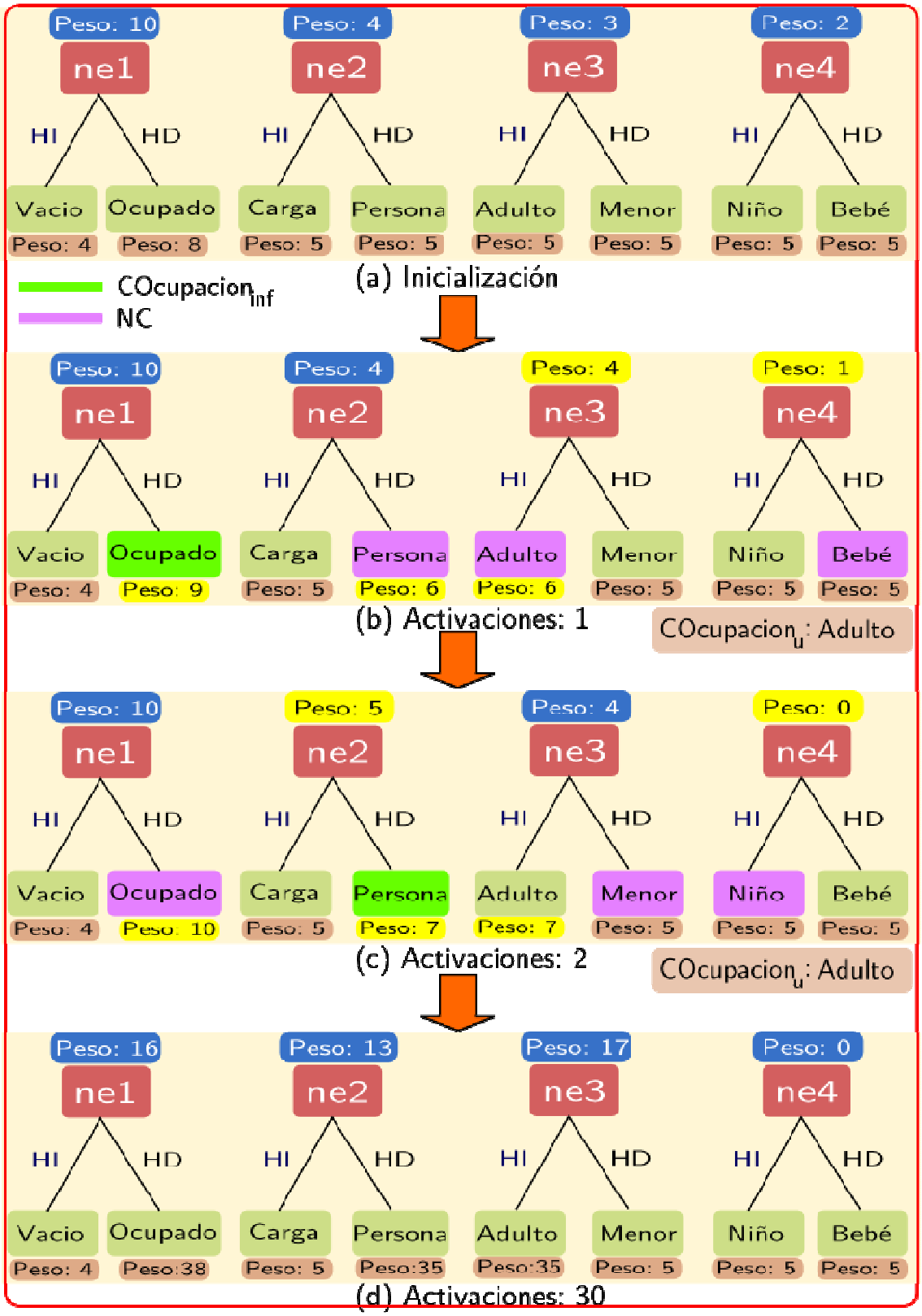



Figura 16. Evolución de la JP de la regla de detección de abrir/cerrar puerta del copiloto para el caso de uso 1

Suponiendo que el usuario siempre indicara adulto como el tipo de ocupación real durante una treintena de veces consecutivas, el estado de la JP quedaría como muestra la Figura 16 (d). En ella vemos que los HD desde NE 1 hasta NE 2, así como el nodo adulto en el NE 3 han incrementado su peso en cada una de las activaciones. De esta forma, existen muchas probabilidades de que o bien adulto o bien alguno de los *COcupacion* en su misma rama de la taxonomía (ocupado o persona) resultasen los elegidos como salida del sistema. Además, vemos que la distribución de los pesos se ha repartido entre los tres primeros NEs por lo que la JP tiene ahora más posibilidades de devolver *Ocupaciones* por encima del NE 1 que constituía su NE por defecto en la inicialización.

Por tanto, en este ejemplo se ha visto que el sistema ha sido capaz de adaptar la JP de una regla a las indicaciones del usuario, otorgando más peso al *COcupacion* que el usuario ha ido indicando repetidamente que era la ocupación real, pero también a los *COcupacion* en la misma rama que lo indicado por el usuario. Además, ahora el JP es más *atrevida* a la hora de realizar inferencias con niveles de especificidad más altos al ganar peso los NE 2 y 3 respectivamente.

#### 6.5.3.2 Caso de uso 2

El caso de uso anterior muestra la situación irreal en la que en un departamento del habitáculo siempre existe el mismo tipo de ocupación. Sin embargo, en la mayoría de ocasiones el tipo de ocupación en un habitáculo variará de un tramo a otro dentro de un mismo itinerario, bien porque alguien o algo se ha bajado del vehículo o bien porque, al contrario, se ha subido.

Este segundo caso de uso pretende ilustrar cuál sería el comportamiento del sistema frente al dicho tipo de situaciones. Al igual que en el caso de uso anterior, suponemos que el usuario simplemente realiza como actividad de carrocería la abertura y cierre de las puertas, aunque en este caso la ocupación del departamento del copiloto alterna los valores de niño y bebe de un tramo a otro. Por tanto, y asumiendo de nuevo la existencia de una regla en un EPA de nivel 1 que se activara cada vez que se abriera y cerrara la puerta del copiloto, la Figura 17 muestra la evolución de la JP de dicha regla a lo largo de sus sucesivas activaciones.

De nuevo la inicialización de la JP es idéntica a la mostrada en el caso de uso 1, tal y como se muestra en Figura 17 (a). En la primera activación de la regla (Figura 17 (b)) el sistema infiere como salida niño para el departamento del copiloto, aunque posteriormente, cuando el usuario corrige el habitáculo inferido, indica que la

ocupación real en dicho departamento era bebé. Pese a que *COcupacion*$_{inf}$ es diferente a *COcupacion*$_{u}$ ambas están al mismo nivel de especificidad (caso B de las situaciones descritas en la sección 6.5.2.2 apartado 3)). Por lo tanto, el NE 4, que es el NE al que pertenecen tanto bebé como niño, no es fiable a la hora de realizar inferencias, por lo que debe ser penalizado. Además, se penaliza también el *COcupacion* niño para que bebé, que ha sido el que el usuario ha indicado, cobre más fuerza para ser elegido en el futuro. A continuación, se recompensa a los HD de todos los niveles inferiores a 4 ya que representan a los *COcupacion* dentro de la taxonomía que están en la rama camino hacia niño. Finalmente, se premia al NE 3 ya que ha sido el NE más alto por debajo del NE 4 cuyo HC estaba en la misma rama que *COcupacion*$_{u}$.

La segunda vez que la regla se activa (Figura 17 (c)) la situación es parecida a la anterior, aunque en este caso la salida del sistema fue el *COcupacion* bebé, cuando el usuario ha indicado niño. En consecuencia, las tareas a realizar con el objetivo de afinar los pesos de la JP son las mismas que cuando la regla se activó por primera vez, salvo que en esta ocasión es el *COcupacion* bebé el penalizado en favor de niño, y el NE 2 el recompensado.

Finalmente, después de una treintena de activaciones la distribución de los pesos de la JP quedaría como muestra la Figura 17 (d). En ella vemos, que el NE 4 ha perdido todo su peso, por lo que las probabilidades que alguno de sus dos hijos salgan elegidos como salida del sistema son muy bajas. Esto se debe a que el sistema detecta que hay mucha variabilidad en el tipo de ocupación a ese nivel por lo que decide no arriesgarse y engordar los pesos de los NE inferiores, así como los de sus HD.

Con este ejemplo, se ha visto cómo el sistema, al detectar muchos errores a un determinado NE, decide penalizar dicho nivel con el objetivo de que resulte poco probable utilizar sus inferencias, y premiar a los *COcupacion* que van camino hasta dicho NE, como son ocupado, persona y menor. Así, aunque el sistema no infiera los *COcupacion* niño o bebé, sí que es muy probable que genere como salida cualquiera de los otros tres *COcupacion* que están en su rama, lo que, pese a que no están al mismo NE que indica el usuario, no constituye una inferencia errónea.

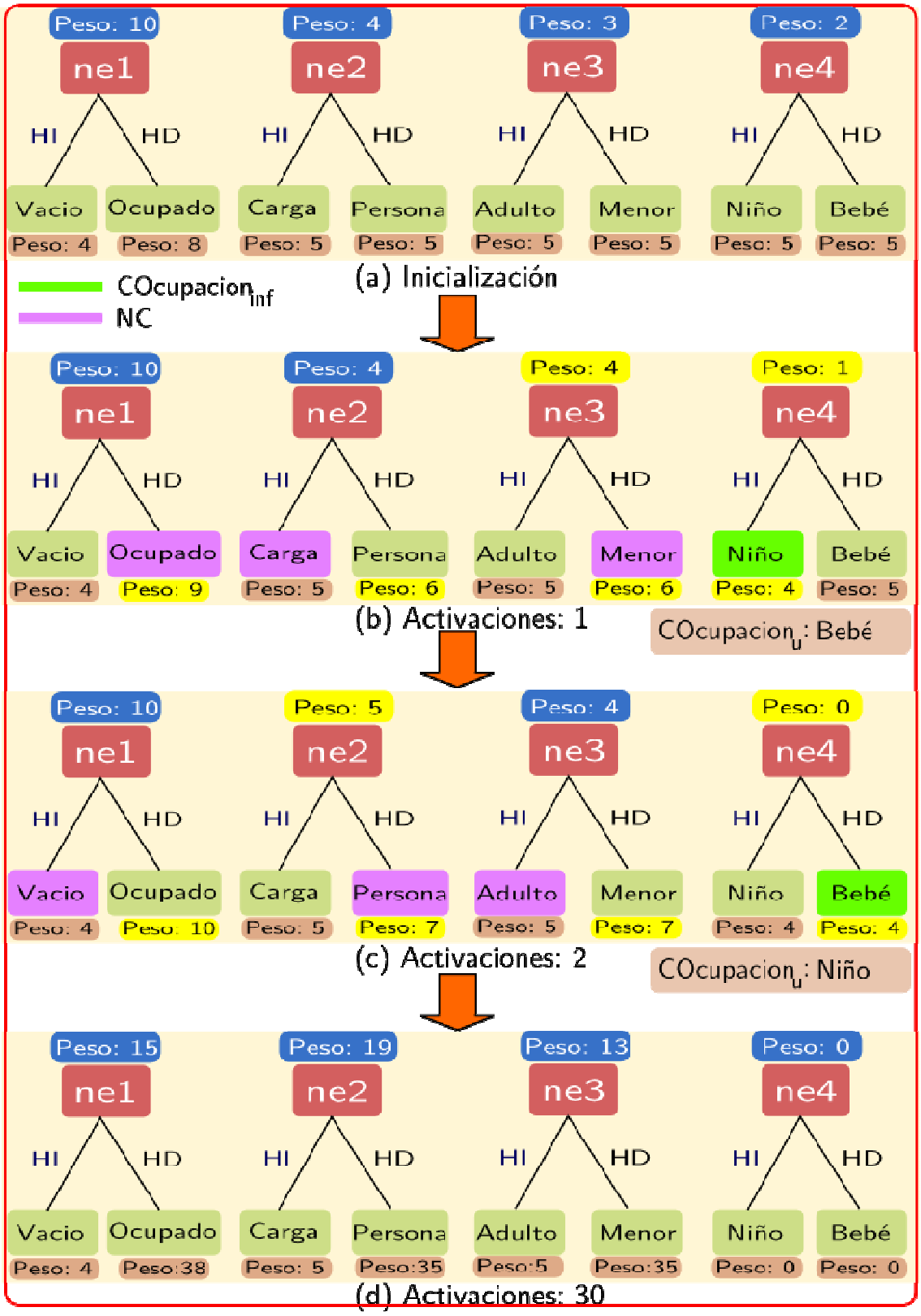



Figura 17. Evolución de la JP de la regla de detección de abrir/cerrar puerta del copiloto para el caso de uso 2

## 6.6 MSAPU

En cada ejecución de MMUC, éste va afinando los pesos de las JPs para cada usuario, tal como se vio en el apartado anterior. Cada vez que la aplicación se cierra, las diferentes JPs deben almacenarse en el perfil de usuario para, que más tarde, el sistema sea capaz de recordar lo que ha aprendido de él. Cada vez que se ejecuta la aplicación de nuevo, se recuperan las JPs del fichero de perfil del usuario y se cargan en el MMUC para que las inferencias que haga el sistema sean en función de los hábitos de ese usuario concreto.

El módulo encargado de realizar la tarea de almacenar y recuperar el perfil de cada usuario es el MSAPU. Concretamente, dicho módulo recupera la información de las JP del perfil del usuario que haya arrancado la aplicación y se los pasa al MMUC. Una vez que el usuario cierre la aplicación, el MSAPU obtiene los JPs del MMUC y los almacena de nuevo como parte del perfil del usuario.

## 6.7 Esquema recopilatorio del diseño

A modo de resumen de todo lo explicado en esta sección la Figura 18 muestra las interacciones de los módulos del sistema entre sí.

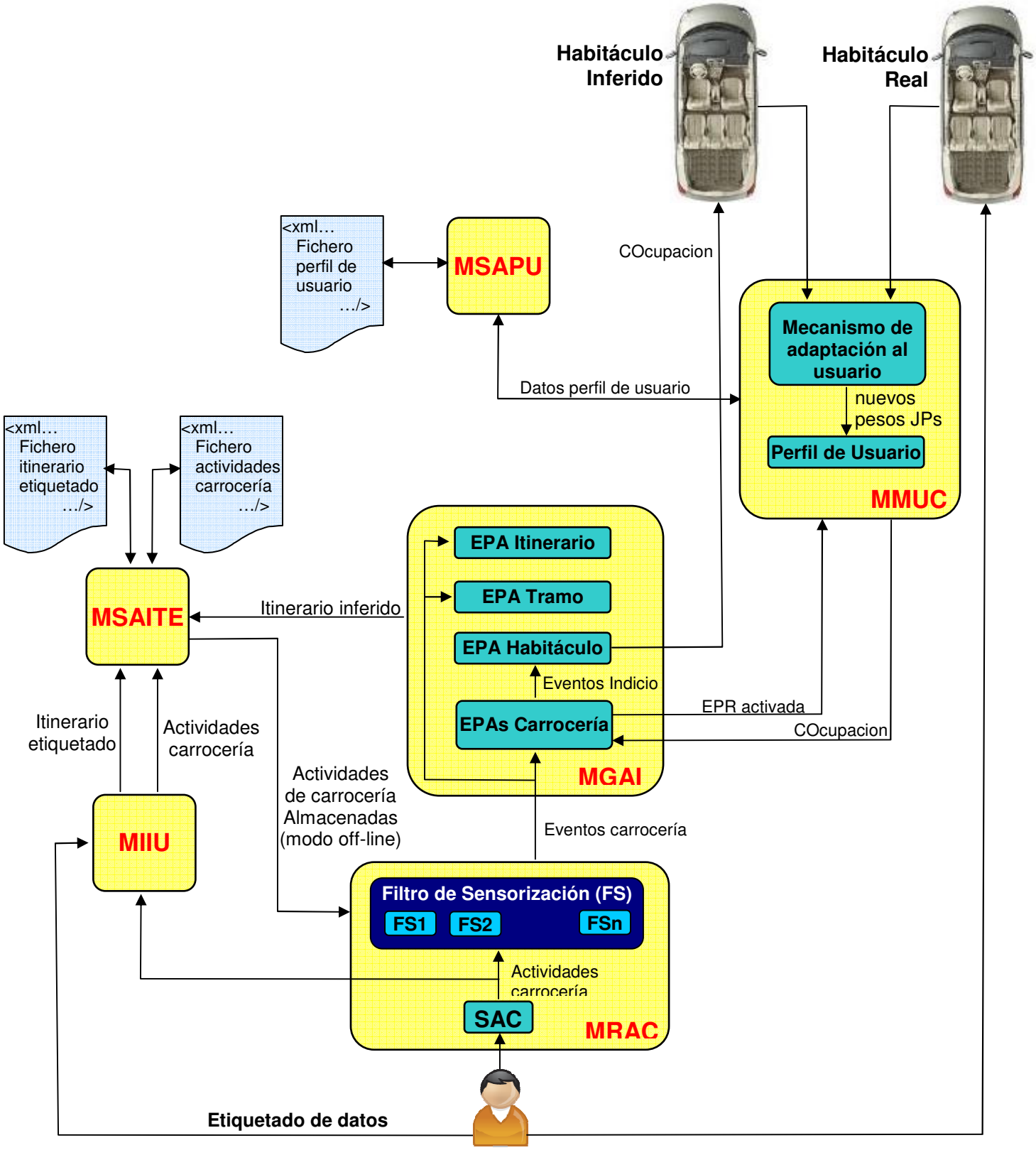



Figura 18. Esquema detallado de las interacciones entre los módulos del sistema

# 7 Implementación del Sistema

Una vez expuestas las decisiones de diseño a la hora de desarrollar el sistema propuesto, el siguiente paso a realizar consistió en implementar todos los módulos definidos. Por lo tanto, en la presente sección se describen los puntos más importantes de la implementación de las diferentes funcionalidades del prototipo.

## *7.1 Plataforma y lenguaje*

La aplicación ha sido desarrollada en el lenguaje de programación Java versión 6. Bajo el entorno de programación *Netbeans IDE 6.8 [26]*.

## *7.2 MRAC y MIIU*

Como se ha comentado en la sección 6.1.3 del presente trabajo, se tomó la decisión de diseño de generar los diferentes valores de los sensores del vehículo a través del SAC y del FS. Debido a que el usuario que introducirá las actividades en el SAC (que después se transformarán en eventos) es el mismo que etiquetará los itinerarios, tramos y ocupaciones a través del MIIU, se decidió implementar un único simulador que contemplará el diseño tanto del SAC como del MIIU. Dicho simulador se ha denominado Simulador para la Generación de Itinerarios y Actividades (SGIA).

Para la realización de la interfaz gráfica de dicho simulador se ha hecho uso de las librerías *swing* y *awt,* explotando la facilidad de desarrollo que ofrece Netbeans para las interfaces gráficas.

### 7.2.1 Simulador para la Generación de Itinerarios y Actividades (SGIA)

Desde este simulador se podrán generar itinerarios supervisados por el usuario correspondientes al MIIU y también las actividades del MRAC que posteriormente constituirán la entrada al MGAI.

Hay varias formas de acceder al sistema según los modos de funcionamiento descritos en la sección 6.1.6. De este modo, el sistema tiene dos modos de ejecución. Uno *off-line*, en el que se generaban los datos del itinerario pero no se procesaban por el MGAI, si no que, por el contrario, se almacenaban para un procesamiento posterior. Otro, el *on-line*, en el que a la vez que se generaban las actividades se enviaban al motor CEP. La decisión de en cual de los dos modos el usuario quiere ejecutar el simulador se debe tomar en la primera pantalla de la interfaz gráfica.


Figura 19. Pantalla de entrada a la aplicación

Las opciones que se tienen para entrar al sistema se explican a continuación en el mismo orden en el que aparecen en la Figura 19:

a) Off-line$_1$: El usuario podrá introducir todos los datos de un itinerario, los tramos de los que está compuesto y la ocupación y actividades de carrocería que se producen en cada uno de los tramos. El itinerario generado se almacenará en un fichero XML.

b) Off-line$_2$: El usuario podrá seleccionar un fichero de un itinerario ya generado para que se procesado en ese momento por el MGAI.

c) On-line: Esta opción es la unión de las dos anteriores. Primero el usuario introducirá los datos de un itinerario y seguidamente será procesado por el motor CEP para realizar inferencias.

Si el usuario arranca las opciones a) o c) se iniciaría el denominado SGIA. Dicho simulador va mostrando al usuario diferentes pantallas donde se irá recogiendo toda la información del itinerario. Primero, se introducirán los datos generales del mismo y, una vez confirmados, se irán definiendo uno a uno los tramos en los que esté compuesto.

Para etiquetar itinerarios, la primera pantalla que se muestra es la indicada por la Figura 20.


Figura 20. Pantalla de datos de itinerario

Dicha pantalla pide diferentes datos que servirán para identificar el itinerario basándose en la tupla descrita en la sección 6.2.1.

- Identificador del Itinerario: Este nombre aparecerá en el nombre del fichero XML, generado en el MSAITE, para su fácil identificación.
- Número de Tramos: Se establece el número de tramos que van a componer el itinerario.
- DNI e Identificador de Usuario: Este dato es necesario para la gestión de los perfiles de usuario.

Una vez introducido el número de tramos, se indican sobre el mapa los puntos de inicio cada tramo, de esta forma se recogen sus coordenadas GPS. Cada punto puede definirse como *Punto exacto* o *Zona*, los cuales hacen referencia a si el punto de partida es un punto GPS exacto, es decir, un garaje o plaza de parking, o si por el contrario, será una zona acotada como cuando se busca aparcamiento por una zona.


Figura 21. Distribución de tramos en el mapa

Como ya se ha explicado, el concepto de itinerario recoge desde que se sale del origen, hasta que se regresa, es decir, es circular, siempre se vuelve al mismo sitio, con lo que la ruta indicada en la pantalla anterior acaba de nuevo en el punto 1.


Figura 22. Ruta de un itinerario

Una vez completado todos los datos generales del itinerario se debe describir cada uno de los tramos en detalle.

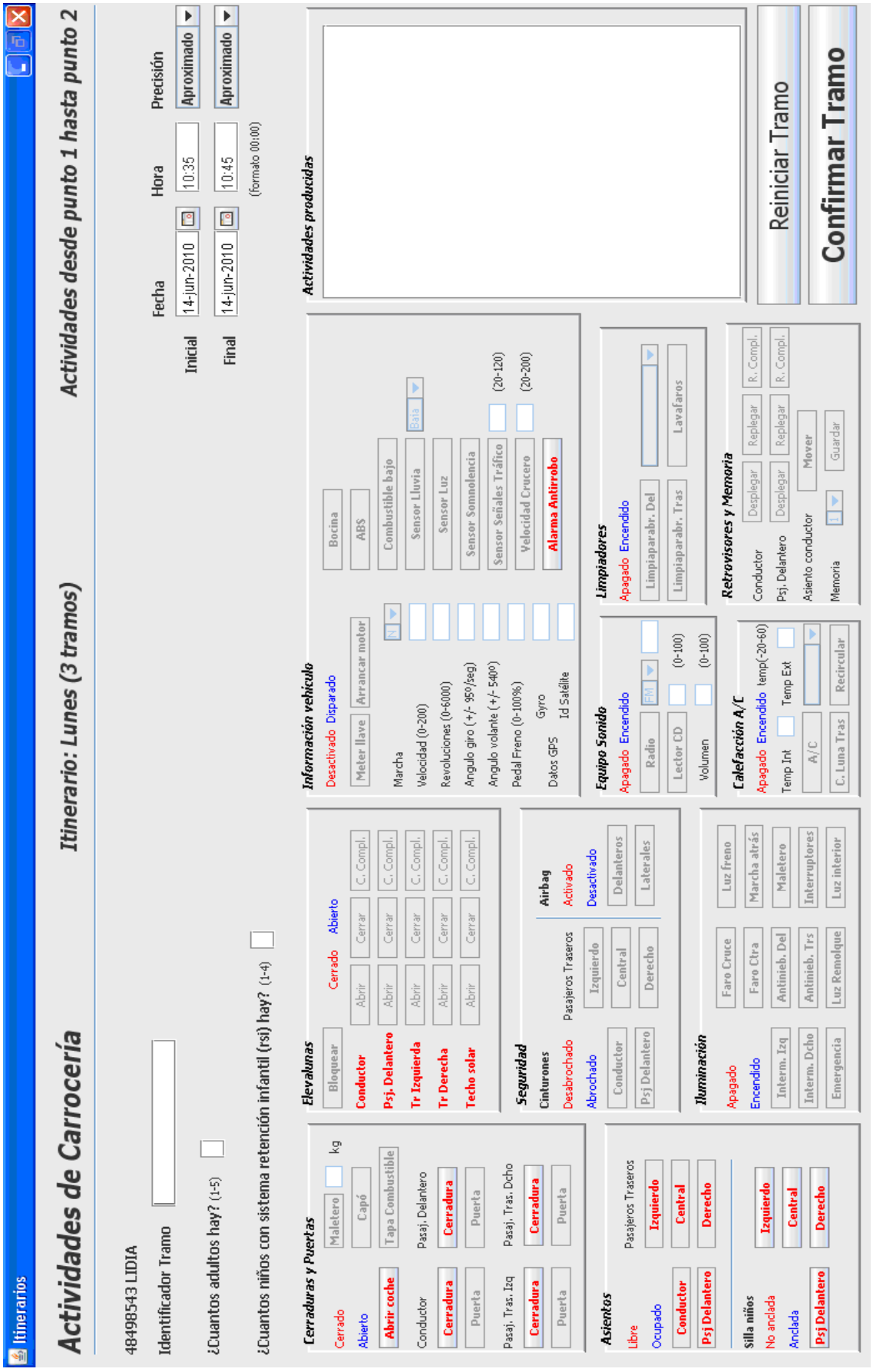



Figura 23. Pantalla de Tramo y Actividades de Carrocería

La pantalla mostrada en la Figura 23 aparecerá para cada tramo que se haya definido en el itinerario. Se pueden apreciar en ella tres secciones principales. La primera de ellas está destinada a introducir los datos identificativos del tramo. La segunda sección, con gran variedad de botones, permite introducir las actividades del mundo real que se pueden hacer en un vehículo. Los botones están agrupados por funcionalidad, ya que como puede apreciarse, hay un panel destinado a realizar acciones de cerraduras y puertas, mientras otro recoge todas las actividades referentes a la iluminación del coche, etc. La tercera sección contiene un área de texto donde se puede ver un histórico de las actividades generadas en el tramo.

Para cada tramo se especificarán los datos contenidos en su tupla identificativa (ver sección 6.2.2):

- Identificador del Tramo: Nombre descriptivo de manera que el usuario lo puede identificar con facilidad.
- Número de Adultos en el habitáculo: Una vez que se introduzca el número de adultos aparecerán una serie de desplegables (tantos como adultos se han indicado) en los que se deberán especificar el departamento del habitáculo que ocupan cada uno de ellos.

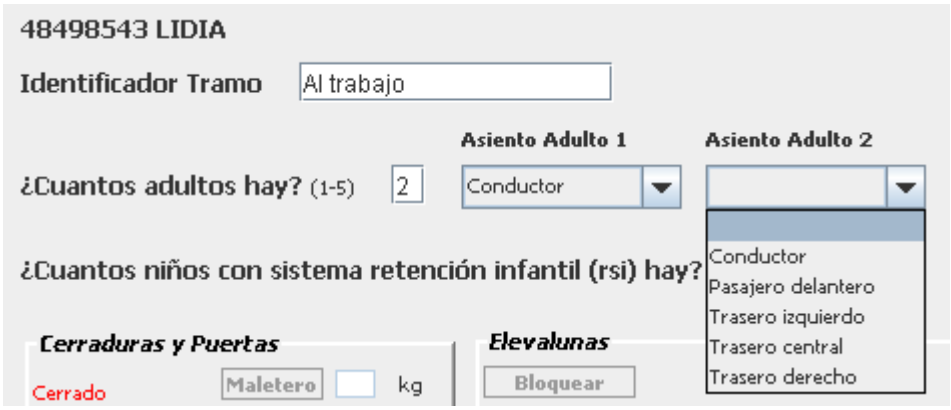



Figura 24. Datos de ocupación

- Número de Niños con RSI en el habitáculo: Una vez que se introduzca el número de niños para los que debe haber un sistema de retención infantil, aparecerán una serie de desplegables (tantos como niños se han indicado) en los que se deberán especificar el departamento que ocupan cada uno de ellos. El sistema comprobará que se haya definido una sillita de niño para ese departamento, en caso contrario no dejará establecerlo, previamente deberemos configurar una sillita para ese sitio.

- Instante inicial y final del Tramo: En que se indica la fecha, y precisión (exacta o aproximada). Por defecto se muestra la fecha y hora actual para el inicio y 10 minutos más tarde para la hora final. El sistema hace validaciones cada vez que se introduce un dato, de manera que no se puedan producir errores y meter una fecha final anterior a la inicial.

Una vez establecidos los datos iniciales debemos indicar todas las actividades de carrocería que ocurren dentro de ese tramo. El SGIA tiene una serie de restricciones intentando simular el funcionamiento real de un automóvil. Se puede observar en la Figura 23 que hay muchos botones deshabilitados, que no se podrán pulsar hasta que no se hayan hecho antes actividades previas. Lo primero que se deberá hacer será pulsar el botón asociado a la actividad de "*Abrir coche*" o bien, cualquiera de sus cerraduras, esto permitirá poder abrir una puerta, más tarde meter la llave, más tarde arrancar motor, etc.

Cada botón o etiqueta cambia de color para indicar el estado en el que está, de esta forma se puede tener una idea del estado del vehículo con solo observar el simulador.


Figura 25. Ejemplo de botones de actividades

La Figura 25 muestra dos de los paneles del simulador, el relativo a los elevalunas y los cinturones de seguridad y airbag. Se puede saber que la ventanilla del conductor se encuentra abierta y que los cinturones del conductor y pasajero delantero están abrochados.

En el área de texto que hay a la derecha de la pantalla se va reflejando toda la actividad que se va teniendo con el simulador, muestra información de tres tipos:

- [INFO]: Muestra información del tramo que se está introduciendo.
- [ERROR]: Informa de cualquier error que esté ocurriendo o actividades no permitidas
- [ACT]: Actividades de carrocería que están ocurriendo y el valor de sus propiedades.

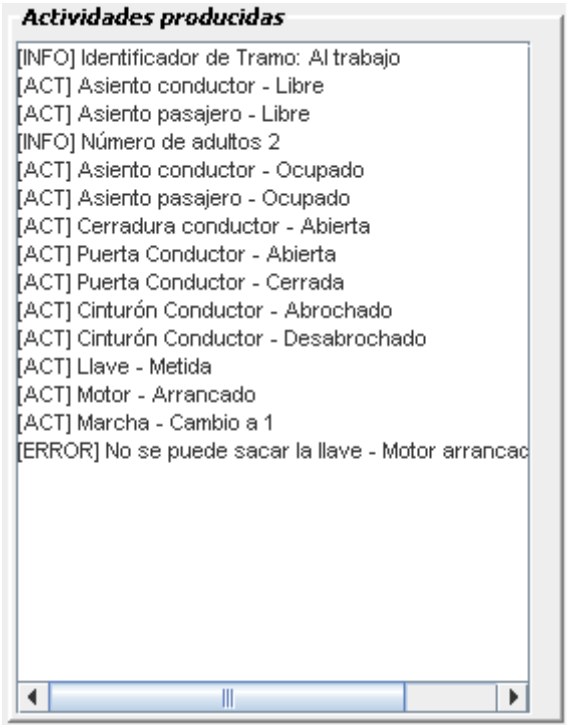



Figura 26. Histórico de actividades producidas

La SGIA permite reiniciar el tramo si se ha introducido alguna actividad errónea, lo cual reestablece el estado inicial del tramo y permite volver a introducir las actividades para el mismo desde el principio.

Una vez se hayan especificado todas las actividades que ocurren para un tramo, se podrá pasar a introducir los datos del siguiente pulsando el botón "*Confirma Tramo*". En este sentido, el simulador no dejará realizar esta acción hasta que no se hayan introducido todos los datos obligatorios.

Como se ha descrito anteriormente, cada uno de los botones del simulador representan una actividad. Las actividades que este módulo soporta fueron las

detalladas en la sección 6.1.4. Por tanto, se ha definido una jerarquía de clases que representan las distintas actividades que se muestra en la Figura 27:

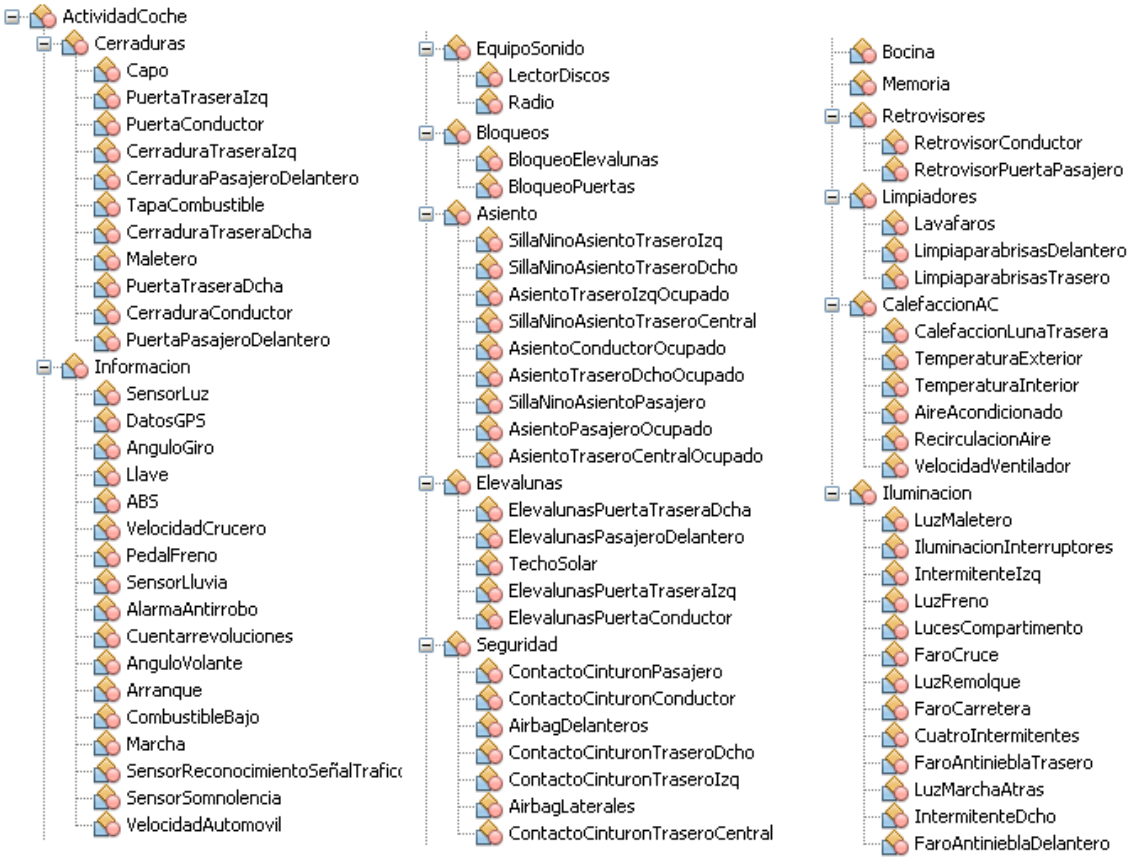



Figura 27. Diagrama de clases de actividades de carrocería

Cuando se pulsa un botón del simulador que representa a una actividad, como consecuencia se crea una nueva instancia de la clase a la que representa y pone sus atributos con los valores adecuados.

Una vez se ha concluido todo el proceso de introducción del itinerario, se podrán hacer dos acciones distintas en función de la opción seleccionada en la pantalla inicial del sistema. Una será la serialización del itinerario introducido, en caso de estar en un modo de ejecución off-line. La otra será recoger únicamente las actividades generadas (sin estructurar en tramos ni itinerarios) y pasarlas al FS que generará los eventos pertinentes para mandarlos al MGAI.

### 7.2.2 Filtro de sensorización (FS)

El filtro ha de seleccionar sólo algunas de las actividades que se han producido en el SGIA para pasarlas al MGAI. Para la implementación del FS se ha desarrollado un EPA cuya implementación se explica en la sección 7.4.3.2.

## *7.3 MSAITE*

El objetivo principal de este módulo es el almacenamiento y persistencia tanto de los itinerarios generados por el usuario a través del SGIA, como de todas las actividades de las que está compuesto cada itinerario sin estar agrupadas en tramos.

### 7.3.1 La clase `Gestor XML`

Se ha desarrollado una clase `GestorXML` que se encarga de toda la gestión de almacenamiento y recuperación de los ficheros XML que se tratan en el presente módulo.

Esta clase, para almacenar todos los ficheros que se generan, crea un directorio *DATOS* a partir de la raíz del sistema. Dentro de este directorio se encontrará un subdirectorio distinto por cada uno de los usuarios (el cual lleva por nombre el DNI del usuario) que contendrá los ficheros XML tanto de itinerarios como de actividades generados por el usuario.

Son varios los puntos de la aplicación desde donde se invoca a `GestorXML`, los cuales se irán comentando conforme se vea en esta sección qué ficheros se han de generar y con qué estructura.

### 7.3.2 Serialización de itinerarios

Una estructura de itinerario puede generarse desde distintos puntos de la aplicación, concretamente:

- desde la SGIA, a partir de los datos que introduce el usuario, y
- desde el MGIA, a partir de las actividades de carrocería que llegan a este módulo se irá generando automáticamente un itinerario por medio de sus inferencias.

Desde los puntos de la aplicación arriba mencionados se invocará a la clase `GestorXML` definida pasándole la instancia del objeto itinerario que se desea serializar.

Se muestra en la Figura 28 un ejemplo XML resultante al serializar una instancia de itinerario por parte de `GestorXML`.


Figura 28. Estructura XML de un itinerario

Dicho itinerario se compone de dos tramos aunque únicamente se muestra en la Figura 28 detalles del primero.

### 7.3.3 Serialización de actividades de carrocería

Las actividades de carrocería se obtienen cuando un usuario introduce un itinerario desde la SGIA. De cada actividad generada se crea dos instancias, una de ellas se almacena como parte de un tramo del itinerario, mientra que la otra se introduce en un listado donde al final se tendrán todas las actividades generadas a través del SGIA sin ningún tipo de agrupación.

Este último listado de actividades se deberá almacenar en un fichero XML que más tarde podrá ser recuperado y servir como entrada al MGIA. Para las tareas de almacenamiento y recuperación del fichero se invoca a la clase `GestorXML` desde la SGIA. Concretamente la recuperación del mismo se realiza cuando se lanza la aplicación en modo off-line[2].

Se muestra en la Figura 29 un ejemplo del formato del fichero XML que contiene únicamente las actividades generadas por el SGIA:


Figura 29. Estructura XML del listado de actividades de carrocería

Esta estructura XML tiene un elemento raíz *Actividades* que contiene un elemento distinto por cada una de las actividades que componen el listado. Como se puede observar en la Figura 29, cada actividad tendrá unos atributos distintos en función de su naturaleza y características. Por ejemplo *ContactoCinturonTrasero* tiene un atributo *activo* que indicará si está o no anclado, mientras *VelocidadAutomovil* tiene un elemento *kmh* que indica los kilómetros por hora a los que circula el coche. Todas tienen en común un atributo *timeStamp* que indica el momento exacto en el que se produjo la actividad tomando como referencia el instante en que se inició el tramo al que pertenecen.

La misma estructura que se observa para el fichero XML de actividades de carrocería es la que tiene el fichero de itinerario para su elemento *actividades*, que se ha mostrado sin desplegar en la Figura 28.

### 7.3.4 Herramienta para tratamiento XML. XStream

Para el tratamiento y gestión de los ficheros XML descritos arriba, la clase `GestorXML` ha de utilizar un *parser* que permita leer documentos XML y trabajar con ellos desde Java.

Hay varias maneras de procesar XML con Java, entre las se pueden destacar las tecnologías SAX, DOM y Digestor. Cada una tiene su propósito. Así, SAX es extremadamente eficiente en la lectura de archivos. DOM proporciona un mejor control de la estructura jerárquica de los documentos XML, siendo más adecuado para escribir documentos para los que nadie tiene el control total de la estructura (esquema). Por último, Digestor ofrece una interesante forma de asignación de documentos XML a objetos Java y viceversa. Sin embargo, requiere que la aplicación se configure manualmente. XStream [27] es otra forma que permite a través de Java realizar puentes-XML.

Finalmente se optó por utilizar XSteam, una API para serializar objetos a archivos XML y luego volverlos a instanciar en base al archivo. Es extensible por el usuario y soporta distintos tipos de parsers como SAX o DOM. Sus principales características son:

- No es necesario configurar el mapeo de forma manual ya que XStream encuentra dinámicamente los atributos de una clase Java y sus respectivos valores. Serializa también los campos internos, incluidos los privados y los estáticos. Incluso soporta referencias circulares y objetos duplicados.
- No se requiere que los objetos java sigan ninguna norma en particular, ni ninguna modificación.
- La estrategia de transformación es configurable ya que permite registrar una manera particular de representar un tipo de dato en XML.
- Soporte para definición de alias de los atributos de una clase (denominados Annotations).

Suponiendo que se tuviera un objeto java `itinerario` que contuviera la estructura de un itinerario, serializarla a un fichero XML con la estructura mostrada en la sección 7.3.2 usando XStream, es tan sencillo como:

```
XStream xs = new XStream();
xs.autodetectAnnotations(true);   //Para que se detecten anotaciones
File f = new File(directorio, nombFichIti);
xs.toXML(this.itinerario, new FileOutputStream(f));
```

Como se ha detallado en las características de XStream, esta librería permite configurar hasta un grano muy fino la apariencia final del XML.

```
@XStreamAlias("Itinerario")
public class Itinerario {

    @XStreamAsAttribute
    protected String identificador;

    @XStreamAsAttribute
    protected int numeroTramos;

    @XStreamImplicit
    protected LinkedList<Tramo> tramos;

    @XStreamOmitField
    protected long dni;
```

En el siguiente fragmento de código se observan varias *Annotations* de XStream, que se pasan a describir:

- *@XStreamAlias("alias")*: Indica un alias para el nombre de la clase. Esta funcionalidad permite que el nombre del elemento en el fichero XML no contenga el nombre del paquete de la clase.
- *@XStreamAsAttribute*: Indica que ese campo se debe convertir en un atributo del elemento ya que por defecto los campos se convierten en subelementos.
- *@XStreamImplicit*: Indica que cada elemento de la lista se cree como un elemento XML sin que todos ellos tengan un elemento común, colgando todos directamente del elemento principal (la clase).
- *@XStreamOmitField*: Indica que el campo al que hace referencia no se incluya en el XML.

Estas anotaciones se han introducido en varias clases en toda la aplicación.

Otra funcionalidad que aporta XStream de la que también se ha hecho uso son los *Converters*. Estos elementos permiten personalizar el formato de algunos campos de la clase en el XML en el caso de que el formato por defecto que proporciona XStream no se acomode a las necesidades que se tengan. Para definir un *Converter*, hay que crear una nueva clase que implemente su interfaz e implementar sus métodos (`marshal`, `unmarshal` y `canConvert`). En el presente trabajo se han implementado los siguientes:

- `FormatoFechaConverter`: Para que campos de tipo Date, de los que obtenemos la fecha, los muestre con el formato "EEE dd-MMM-yyyy".
- `FormatoHoraConverter`: Para que campos de tipo Date, de los que obtenemos la hora, los muestre con el formato "HH:mm".
- `AsientoConverter`: A este converter le llega una estructura de ocupación compuesta por un HashMap y se parametriza para que la estructura de ocupantes sea tal como se ha mostrado en la Figura 28. A continuación se muestra el código de este converter.

```
public class AsientoConverter implements Converter {

    public void marshal(Object source, HierarchicalStreamWriter writer,
                        MarshallingContext context) {

        HashMap<asientos, Indicio> map = (HashMap<asientos, Indicio>) source;

        for (Iterator it = map.entrySet().iterator(); it.hasNext();) {
            Map.Entry entry = (Map.Entry) it.next();

            Indicio i = (Indicio) entry.getValue();

            writer.startNode("asiento");
            writer.addAttribute("ubicacion", entry.getKey().toString());

            writer.setValue(i.getCOcupacion().getClass().getSimpleName());
            writer.endNode();
        }
    }

    public boolean canConvert(Class type) {
        return type.equals(HashMap.class);
    }
}
```

Una vez definido el *Converter* debe asociarse al atributo que lo va a utilizar mediante la anotación `@XStreamConverter`

```
@XStreamConverter(FormatoFechaConverter.class)
protected Date fecha;
```

De esta manera se ha logrado en el prototipo desarrollado la funcionalidad y manejabilidad de la tecnología XML.

## *7.4 MGAI*

Para implementar la JAE del MGAI, se debe usar una solución CEP/ESP. En este sentido, existen varias herramientas comerciales. Sin embargo, para este proyecto se ha usado la solución de código abierto llamada Esper [28]. Dicha API está implementada en Java y actualmente está disponible bajo licencia GPL.

### 7.4.1 Procesamiento de eventos complejos mediante Esper

Esper permite el procesamiento a gran escala combinando dos mecanismos diferentes, por un lado las ventanas temporales y por otro la ejecución de consultas complejas de forma continua.

Las ventanas temporales son estructuras capaces de retener un número específico de eventos hasta que se cumpla una condición que indique que hay que borrarlos. Por otro lado, las consultas complejas se estructuran en Esper a través de sentencias EPL (Event Processing query Language), que son parecidas a las consultas SQL y sirve para estructurar flujos de eventos.

Esper también soporta otro tipo de consulta a través de los denominados patrones, que permiten especificar un orden concreto en el que se tienen que dar los eventos para que el patrón se cumpla. A través de las consultas EPL se pueden seleccionar diversos datos como combinación de distintos patrones.

Las sentencias EPL se utilizan para agregar información procedente de uno o varios flujos de eventos. Cuando una sentencia EPL se activa es porque el patrón que define se cumple. Para obtener los resultados de la misma se pueden utilizar dos mecanismos: mediante listeners, que reciben información actualizada cada vez que el motor procesa eventos que cumplen su sentencia EPL asociada, o bien mediante suscriptores.

En definitiva Esper es como una base de datos al revés: almacena consultas y pasa el flujo de datos a través de ellas. Sin embargo, su modo de ejecución es continuo sobre el flujo de datos de entrada, al contrario que en una base de datos relacional donde sólo se ejecuta una consulta cuando se lanza.

### 7.4.2 Incorporación de Esper al Sistema

Para incorporar Esper al sistema se ha desarrollado una clase llamada `GestorCEP`, que será la encargada de gestionar todo lo referente al motor CEP de Esper. Esta clase se ha implementado siguiendo el patrón *Singleton* para asegurar que siempre se trabaja con una única instancia. Los pasos que ejecuta dicha clase son mostrados en la Figura 30 y descritos a continuación.

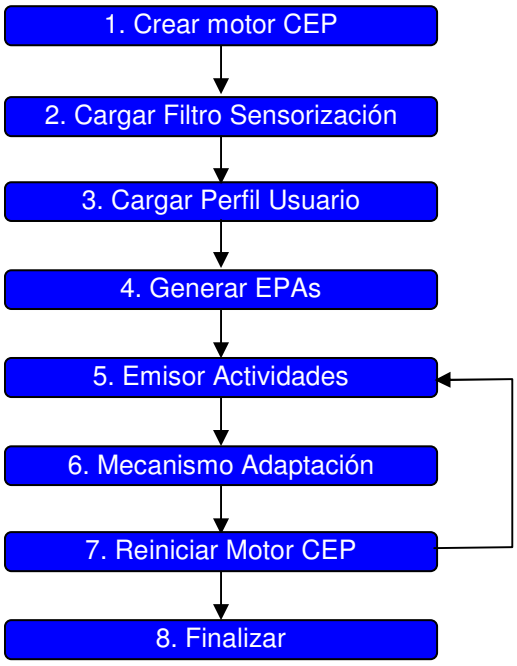



Figura 30. Flujo de ejecución de la clase `GestorCEP`

1) Crear el motor CEP. Instancia la clase que contendrá el motor CEP de Esper. Para ello hay que pasarle como parámetro las clases que ejercerán el papel de eventos.

```
public static EPServiceProvider ep;
(…)
Configuration config = new Configuration();
config.addEventTypeAutoName("confort.actividades");
config.addEventTypeAutoName("CEP.Eventos");
ep = EPServiceProviderManager.getDefaultProvider(config);
```

Añadimos los paquetes que contienen los eventos

En el código arriba mostrado, la variable `ep` representa al motor CEP.

2) Cargar filtro de sensorización. Se debe invocar al EPA que implementa el FS, denominado `EPAFiltroSensorizacion`. Una vez cargado en el sistema, este EPA se queda en ejecución durante todo el tiempo que dure la aplicación escuchando

actividades provenientes de `EmisorActividades` (ver paso 5) y creando eventos a partir de algunas de las actividades, según el tipo de vehículo que se quiera simular. La implementación de `EPAFiltroSensorizacion` se define en la sección 7.4.3.2.

3) Invocar al gestor de la configuración de usuario. Se invoca a la clase `GestorConfUsuario` que cargará el perfil de usuario (tal y como se describe en la sección 7.6). El motor CEP guardará dicho perfil en una variable estática. Ésta será consultada por los EPAs dentro del motor CEP para generar los eventos indicio, ya que dicha variable contendrá las JPs de todos los EPRs.

```
public static ConfUsuario cu;

gcu = new GestorConfUsuario(dni);
gcu.cargarConfiguracion();
```

Carga en ***cu***, la estructura con la Configuración de usuario

4) Invocar al gestor de EPAs. A continuación, se invoca a la clase `GeneradorEPAs`, cuyo cometido es crear la jerarquía de EPAs (ver la Figura 9) en el motor CEP creado en el paso 1. Dicha clase va creando uno a uno cada EPA. La estructura de estos EPAs se verá más adelante en esta misma sección.

5) Invocar al emisor de actividades. El `GestorCEP` recibe las actividades de carrocería pudiendo estas venir en dos formatos distintos. Por un lado, como un archivo XML proveniente del MSAITE, si se está en modo *off-line.* Por otro lado, como una lista de actividades provenientes de la SGIA, si se está haciendo en modo *on-line*. A continuación, el `GestorCEP` pasará las actividades a la clase `EmisorActividades` que se encargará de recorrer las actividades y enviarlas al `EPAFiltroSensorizacion` para que las filtre y genere eventos solo a partir de las actividades pertinentes (según la sensorización del tipo de vehículo).

6) Lanzar el MMUC. Cuando la clase `EPATramo` detecta el final de un tramo, muestra el habitáculo inferido al usuario, pudiendo realizar éste las correcciones que crea pertinentes.

7) Reiniciar el motor CEP. Una vez terminado el paso 6, se modifican los JPs (según se describió en la sección 6.5.2.2 apartado 3) y `GestorCEP` reinicia el motor CEP para volver a lanzar el proceso a partir del evento que ha provocado que se infiera

el final del tramo. `EmisorActividades` empezará de nuevo a mandar eventos al motor a partir de ese último.

8) Finalizar. Cuando `EPAItinerario` detecta el final del itinerario deben realizarse dos funciones:

   a) Invoca a la clase `GestorXML` para almacenar el itinerario inferido por el sistema en un fichero XML.

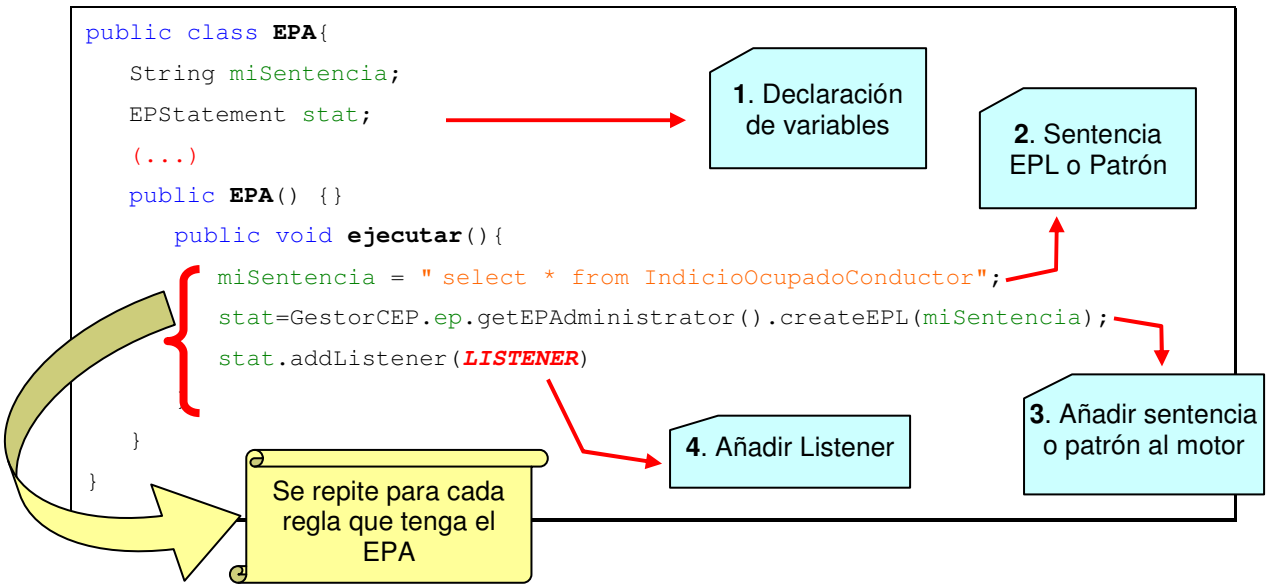
   b) Invoca a la clase `GestorConfUsuario` para que almacene el nuevo perfil del usuario provocado por la modificación de los pesos de las JPs en el MMUC.

### 7.4.3 Los Agentes de Procesamiento de Eventos de la Jerarquía de Abstracción de Eventos

A continuación se exponen los detalles de implementación de los EPAs contenidos en la JAE diseñada.

#### 7.4.3.1 Estructura general de los Agentes de Procesamiento de Eventos del sistema

Cada EPA se ha implementado mediante una clase distinta. Sin embargo, todas tienen una misma estructura común que se describe a continuación:




1) Declaración de variables: En todos los EPAs se podrá encontrar una variable de tipo `String` llamada `miSentencia` que almacenará la sentencia EPL o el patrón asignado al EPA. También se declarará una variable de tipo `EPStatement` propia de Esper que permitirá ejecutar la sentencia EPL o el patrón de `miSentencia` en el

motor CEP. Además de estos atributos cada EPA tendrá sus propias variables específicas.

2) Sentencia o Patrón: Cada regla irá modificando la variable `miSentencia` para definir su propia sentencia o patrón

3) Añadir la sentencia o patrón al motor CEP: Esta línea de código añade la sentencia descrita en el paso anterior al motor CEP.

4) Añadir Listener: Todas las reglas tendrán un listener asociado con la funcionalidad que se ha de ejecutar en caso de activarse la regla. La implementación particular de cada listener se define más adelante.

En la aplicación, según necesidades de cada EPA se han ido creando sentencias EPL o bien patrones. Seguidamente se muestran algunos ejemplos de sentencias o patrones que se han usado en los diferentes EPAs.

Patrón en el `EPAConductorPuerta` que detecta que la puerta del conductor se ha abierto y cerrado posteriormente:

```
every PuertaConductorF(abierta = true) → PuertaConductorF(abierta = false)
```

Para simular la actividad real de abrir la puerta del conductor, habrá que pulsar en el SGIA el botón “Puerta conductor”. Esto recoge la acción del usuario y más tarde el `EPAFiltroSensorizacion` transformará la acción en un evento entendible por el resto de EPAs del motor CEP.

Se puede observar que el patrón comienza por `every`, esto hace que se active, no sólo una vez, sino cada vez que se escuche esa secuencia de eventos. El orden en el que tienen que ocurrir los eventos lo impone el operador→, el cual indica que primero se tendrá que percibir un evento `PuertaConductor` con su atributo `abierta` con valor `true` (indica que la puerta está abierta) y después se deberá escuchar otro evento `PuertaConductor` cuyo valor de `abierta` sea `false` (indica que la puerta está cerrada).

Sentencia EPL en el `EPACopilotoBloqueo` que detecta que, habiéndose inferido hasta el momento la presencia de una persona en el departamento del copiloto, se ha movido la ventanilla y posteriormente se ha bloqueado.

```
select * from pattern [a=IndicioPersonaCopiloto →
                       ElevalunasPasajeroDelanteroF → BloqueoElevalunasF]
```

Esta sentencia está escuchando, por un lado, eventos indicio generados por algún EPA de nivel inferior de la JAE (`IndicioPersonaCopiloto`) y por otro lado, eventos generados directamente por la clase `EPAFiltroSensorizacion` (`ElevalunasPasajeroDelanteroF` y `BloqueoElevalunasF`). Así, combina todo ello imponiendo el orden en el que se tienen que percibir para hacer saltar su listener. En caso de que se haga cierta la regla, el listener operará tal y como se describe en la sección 7.4.3.3.

Sentencia EPL en `EPAHabTrasSRI` que detecta que el cinturón trasero izquierdo estaba anclado antes de que se hubiera abierto ninguna cerradura y el nivel inferior ha detectado un menor en ese departamento.

```
select * from pattern [ContactoCinturonTraseroIzqF(activo=true) →
                       (CerraduraConductorF(abierta=true) or
                        CerraduraPasajeroDelanteroF(abierta=true) or
                        CerraduraTraseraIzqF(abierta=true) or
                        CerraduraTraseraDcha(abierta=true) ) →
                       a=IndicioMenorTrasIzq]
```

Al igual que la sentencia explicada anteriormente, en esta se combinan varios tipos eventos, por un lado los eventos indicio de un EPA de nivel inferior y por otro lado, eventos generados gracias al `EPAFiltroSensorizacion`. En caso de activarse la regla, se generará un `IndicioBebeTrasIzq`, ya que supone que el coche tiene una silla de niños anclada en ese departamento.

### 7.4.3.2 Agente de Procesamiento de Eventos del Filtro de Sensorización

Para la implementación del FS se crea la clase `EPAFiltroSensorizacion`, encargada, de transformar actividades en eventos y enviar los mismos al motor CEP de Esper. Pero no todas las actividades que emite `EmisorActividades` han de ser transformadas en eventos, sino sólo aquellas que se adapten al perfil del vehículo que se quiere simular.

La clase `EPAFiltroSensorizacion` sigue la misma estructura general descrita en el apartado anterior. Para la consecución de su objetivo se creará una sentencia EPL por cada una de las actividades que se quieran transforma en eventos. La metodología empleada para la generación de los nuevos eventos será renombrar las actividades

añadiendo al final de su nombre el carácter "*F*", el cual indica que es una actividad filtrada, ésta se introduce en el motor CEP de Esper a través de la siguiente sentencia:

```
miSentencia =
   "INSERT INTO CerraduraConductorF SELECT * FROM CerraduraConductor";
stat = GestorCEP.ep.getEPAdministrator().createEPL(miSentencia);
```

La sentencia anterior toma la actividad `CerraduraConductor` proveniente de `EmisorActividades` y crea a partir de ella el evento `CerraduraConductorF` insertando éste último en el motor CEP de Esper. El nuevo evento generado será entendible por el resto de EPAs de la aplicación.

Habrá que crear tantas sentencias de este tipo como actividades se quieran filtrar. Así, un vehículo poco sensorizado tendrá en su `EPAFiltroSensorizacion` muy pocas sentencias de este tipo, mientras que un vehículo más sensorizado tendrá un mayor número de ellas.

### 7.4.3.3 Agente de Procesamiento de Eventos de Carrocería

La estructura general de cualquiera de los EPAs de carrocería concuerda perfectamente con la descrita en la sección 7.4.3.1. Tal como se describe en la estructura general de un EPA, todas las sentencias o patrones de los EPAs desarrollados tienen un listener asociado. Cada vez que un patrón o sentencia EPL se activa, el motor CEP de Esper invoca a su listener asociado. Los listener tienen la misma estructura para todos los EPAs de carrocería de la JAE. Esta estructura sigue los siguientes pasos.

1. Generar una ocupación. Consulta al MMUC qué ocupación se ha de generar por haberse activado la regla mediante la siguiente instrucción:

```
ocupacion =
GestorCEP.cu.getEpas().get(indiceEPA).getReglas().get(regla0).getConsecuente()
```

   Donde `indiceEPA` indica qué EPA estamos tratando y `regla0` indica la regla del EPA. La función `getConsecuente` de la Regla invoca al método de selección descrito en la sección 6.5.2.2 apartado 2).

2. Generar un evento indicio. Una vez que se tiene la ocupación se envuelve en un evento Indicio asignado al EPA al que pertenece la regla:

```
Indicio i = Indicio.generaIndicio(asiento, ocupacion);
```

Donde `asiento` indica el departamento del vehículo. Al nuevo evento indicio generado se le deben añadir a continuación sus causas. Para ello:

a. Se accede a los antecedentes de la sentencia o patrón que ha saltado. Así, si por ejemplo, la sentencia activada es

   ```
   select * from IndicioPersonaConductor
   ```

   su antecedente será `IndicioPersonaConductor`. Esper proporciona un método `getUnderlying` que permite recuperar la instancia completa dado un antecedente en una consulta EPL o patrón.

b. Una vez recuperado el Indicio, se obtienen sus causas y se añaden al nuevo Indicio que se está generando:

```
causas = iNivelInferior.getCausas();
i.setCausas(causas);
```

c. Posteriormente, se le añade al nuevo indicio el EPA y la regla actual como la causa más reciente.

```
i.añadeCausa(indiceEPA, regla0);
```

3. Introducir el nuevo indicio en el sistema. Se introduce dicho evento en el motor CEP para que pueda ser procesado por los EPAs de niveles superiores, lo cual se logra mediante la siguiente instrucción:

```
GestorCEP.ep.getEPRuntime().sendEvent(i);
```

#### 7.4.3.4 Agente de Procesamiento de Eventos Habitáculo

Este EPA se implementa mediante la clase `EPAHabitaculo` que únicamente escucha los eventos Indicio que se han ido generando en la JAE. La estructura de la clase que implementa el EPA es igual que la descrita para los EPAs de carrocería, sin embargo, el listener asociado a él tiene una estructura totalmente diferente.

Su misión principal es la de ir escuchando los eventos Indicios que los EPAs de carrocería envía al motor CEP de Esper para más tarde poder generar el habitáculo inferido a partir de esos eventos.

Para cada departamento del habitáculo, pueden llegar indicios que infieran distintos tipos de ocupación. Así, por ejemplo, para el asiento trasero izquierdo puede, en un primer momento, recibir un indicio que contenga como ocupación el valor carga, y posteriormente otro indicio proveniente de otro EPA que contenga como ocupación el valor bebé. Por lo tanto, por cada departamento se mantiene un contador para cada tipo de ocupación para llevar un registro de cuantas veces en un tramo se ha inferido dicho tipo de ocupación para ese departamento.

#### 7.4.3.5 Agente de Procesamiento de Eventos de Final de Tramo

El EPA de final de tramo se implementa mediante la clase `EPAFinTramo` que es la encargada de detectar el final de un tramo. Esto provoca que salten los listener asociados a sus reglas. Estos listener tienen tres funciones:

1) Establecer qué evento ha provocado el final del tramo. Esto se hace porque, como ya se explicó anteriormente, después de la detección de cada tramo se reinicia el motor para que las nuevas inferencias tengan en cuenta los cambios en perfil de usuario.
2) Lanzará un proceso que establecerá el habitáculo inferido con sus ocupaciones correspondientes para cada departamento. Para ello consulta los diferentes contadores mantenidos por el `EPAHabitáculo` por cada ocupación en cada departamento. Así, elige como ocupación de salida de un departamento aquel que tenga el valor en el contador más alto (o lo que es lo mismo, haya sido inferido más veces por el sistema). En caso de empate entre dos o más ocupaciones se seleccionará aquella con mayor nivel de especificidad.
3) Finalmente, el habitáculo inferido es enviado al MMUC.

#### 7.4.3.6 Agente de Procesamiento de Eventos de Itinerario

Se implementa por medio de la clase `EPAItinerario`. Cuando el dicha clase detecta el final del itinerario lanza el paso 8 de la Figura 30.

De momento el sistema soporta la entrada al motor CEP de un único itinerario, con lo que se detectará el final del mismo cuando se dejan de emitir eventos por parte de `EmisorActividades`.

## 7.5 MMUC

Este módulo hace uso de una interfaz gráfica con el fin de mostrar al usuario el habitáculo inferido cada vez que se detecta el final de un tramo para que el usuario pueda corregirlo.


Figura 31. Pantalla de presentación del habitáculo inferido

Una vez que el usuario ve el habitáculo inferido puede, o bien aceptarlo o bien corregirlo. Cuando el usuario lo acepta, confirma estar de acuerdo con todas las inferencias. En caso de que el usuario prefiera corregirlo, se le mostrará la ventana indicada en la Figura 32.

Como se puede observar, para cada departamento del habitáculo se muestra la taxonomía de ocupación en donde por defecto viene marcada la ocupación inferida por el motor CEP. El usuario podrá aquí realizar los cambios que vea oportunos de manera que se refleje ocupación real del coche para el tramo que se está tratando.

Una vez que el usuario ha hecho las correcciones pertinentes, generando así el habitáculo real, deberá pulsar la opción “Guardar”. Esta acción desencadena el paso 6 de la Figura 30 que provoca que el `GestorCEP` invoque a clase `MecanismoAdaptacion`

que es la encargada de lanzar el mecanismo de adaptación definido en la sección 6.5.2.2 apartado 3).

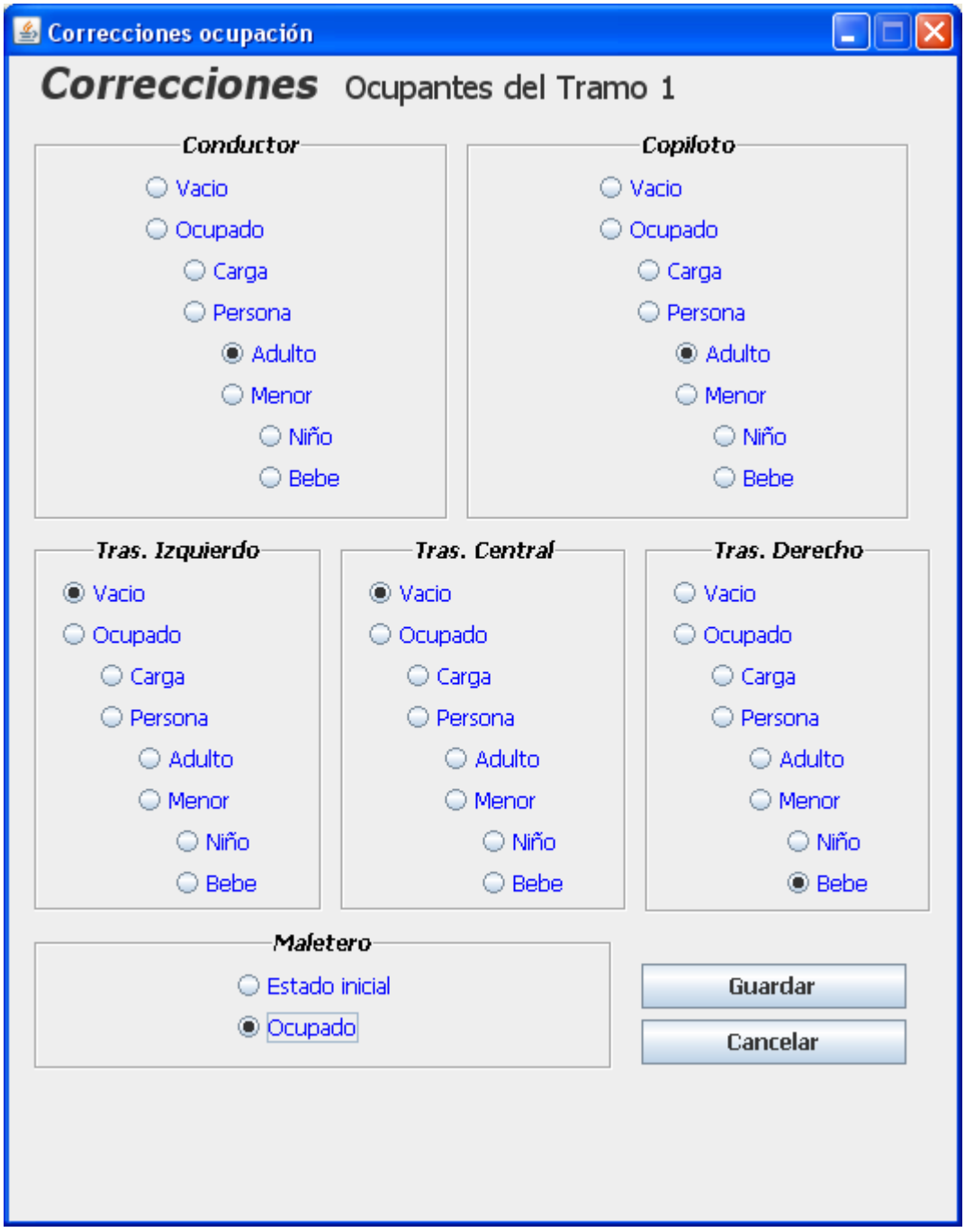



Figura 32. Pantalla para la corrección del habitáculo

## 7.6 MSAPU

El MSAPU ha sido implementado por la clase `GestorConfUsuario`. Dicha clase es la encargada de realizar las principales tareas del presente módulo, como son la

creación, almacenamiento y recuperación de perfiles de usuario por medio de ficheros XML.

`GestorConfUsuario` crea un directorio *CONFIGURACION* en la raíz de la aplicación donde se almacenarán todos los ficheros de perfil de los distintos usuarios del sistema. Así, esta clase busca en un primer momento un fichero XML con el perfil del usuario en el directorio. En caso de éxito, carga el perfil del usuario desde el fichero encontrado usando la tecnología XStream. En caso contrario, genera el perfil por defecto para todo nuevo usuario. En ambos casos el perfil de usuario es almacenado en la clase `ConfUsuario`.

### 7.6.1 La clase `ConfUsuario`

Como se ha dicho, el perfil de usuario se almacena en la clase `ConfUsuario` cuya estructura se muestra en la Figura 33.

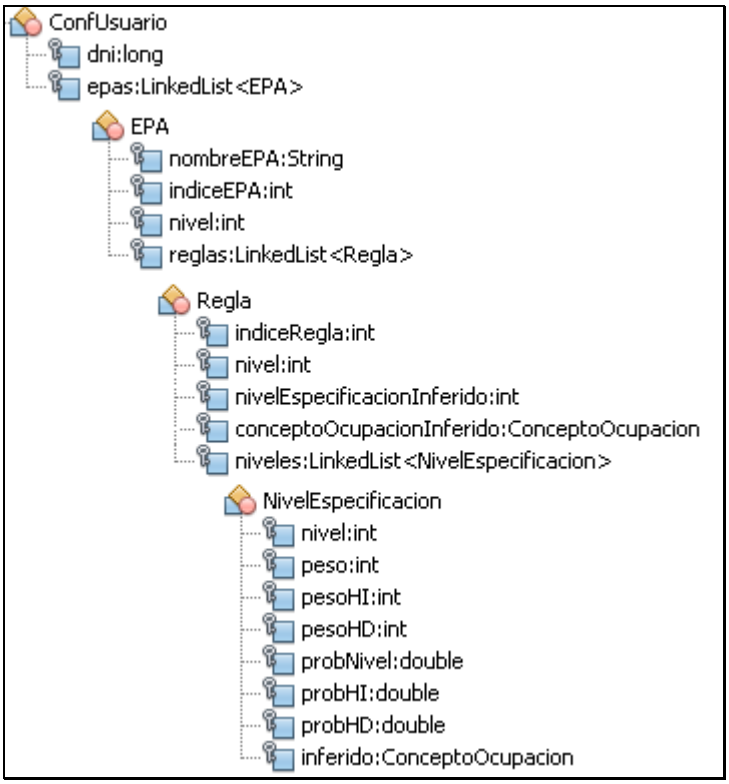



Figura 33. Estructura de clases para representar el perfil de usuario

En la raíz de la estructura se almacena el DNI del usuario, que será la clave de acceso usada por `GestorConfUsuario` tanto para almacenar como para recuperar el perfil. La estructura se compone de un listado de clases `EPA`, las cuales representan cada una a un EPA de carrocería de la JAE. Cada `EPA` almacena a su vez, además de

atributos identificativos, un listado de clases `Regla` que representan a cada uno de los EPR del EPA de carrocería. Cada instancia de `Regla` contiene además atributos como `conceptoOcupacionInferido` y `nivelEspecificacionInferido` que almacenan las últimas inferencias realizadas por ese EPR, datos necesarios para el ajuste de los pesos en la ejecución del mecanismo de adaptación. `Regla` contiene un listado con los cuatro niveles de especificidad que tiene cada EPR representados por la clase `NivelEspecificidad`, la cual, es contenedora de datos necesarios para el proceso de selección descrito en 6.5.2.2 apartado 3).

### 7.6.2 Inicialización del perfil de usuario

Crear un perfil de usuario por defecto consiste en crear una nueva instancia de `ConfUsuario` (Figura 33) e iniciar todos los pesos de las JPs de las reglas con valor 0. Únicamente se iniciará el peso con valor 10 para la ocupación que se considere más lógica en cada regla (las ocupaciones por defecto son las definidas en los distintos niveles de EPAs de la sección 6.4.2), suponiendo esto también dar un valor 10 al peso del NE que contiene el hijo que representa la ocupación.

### 7.6.3 Serialización de perfil de usuario

Al finalizar la ejecución del sistema (paso 7 de la Figura 30), `GestorConfUsuario` se encarga de serializar el perfil del usuario contenido en la instancia de `ConfUsuario` que representa el perfil del usuario que está ejecutando en ese momento el sistema, volcando la información que contiene a su fichero XML mediante XStream.

Dicho fichero almacena el perfil del usuario que viene definido por los pesos de la JP de cada EPR de la JAE. En la Figura 34 se puede ver un ejemplo de su estructura.

Se puede apreciar que la estructura está compuesta por una lista de objetos EPAs, cada uno de estos representa a un EPA de la JAE y contiene a su vez una lista de objetos Regla los cuales representan a cada una de las EPRs del EPA. Cada EPR contiene su JP asociada, incluyendo los pesos de los NE y de sus hijos izquierdo y derecho.


Figura 34. Estructura XML del perfil de usuario

# 8 Experimentación y Resultados del mecanismo de adaptación y consolidación

Se han desarrollado una serie de pruebas para sustentar la calidad del MMUC implementado. Para ello se ha estudiado el comportamiento de las inferencias del prototipo en una serie de simulaciones. Se expone a continuación la planificación de las pruebas realizadas, su implementación y los resultados obtenidos.

## *8.1 Planificación de las simulaciones*

Las simulaciones se han centrado en la inferencia de la ocupación del habitáculo en cada tramo del itinerario, dejando como trabajo futuro el desarrollo de una batería de pruebas para la inferencia de tramos e itinerarios por parte del prototipo.

El objetivo de las simulaciones será, por tanto, comprobar si el prototipo es capaz de adaptarse al "comportamiento del usuario". Por ejemplo, para un usuario que no se pone el cinturón de seguridad, el sistema únicamente escuchará los eventos abrir puerta y cerrar puerta, presuponiendo por defecto que la ocupación del departamento del habitáculo es una carga. El usuario, a través del MMUC, podrá corregir la inferencia del prototipo estableciendo que la ocupación realmente es un adulto en vez de una carga. El sistema deberá entonces tener en cuenta que para ese usuario, con únicamente escuchar que la puerta se abre y se cierra, deberá inferir adulto.

El modo de realizar las simulaciones será repetir la introducción de un mismo itinerario un determinado número de veces, en concreto se repetirá durante 100 iteraciones, para ver cuando las inferencias del prototipo convergen a la ocupación real. Para las simulaciones siempre se parte de un perfil de usuario por defecto que se irá adaptando durante las 100 iteraciones de la simulación.

Dos han sido los escenarios estudiados, los cuales se pasan a describir.

### 8.1.1 Escenario 1

Este primer escenario se centrará en el departamento trasero derecho del habitáculo, donde dos usuarios distintos llevan un niño sentado, realizado cada uno de ellos distintas actividades de carrocería para poner al niño en el departamento indicado. Estas actividades se muestran en la siguiente tabla.

| **Departamento** | Trasero Derecho | |
|---|---|---|
| **Ocupación** | ***Niño*** | |
| **Actividades** | USUARIO 1 | USUARIO 2 |
| | 1. Abre puerta<br>2. Pone cinturón seguridad<br>3. Cierra puerta<br>4. Movimiento de ventanilla<br>5. Bloqueo de ventanillas | 1. Abre puerta<br>2. Pone cinturón seguridad<br>1. Cierra puerta |

Tabla 2. Actividades del escenario de simulación 1

En este caso, se comprobará si el sistema es capaz de realizar la misma inferencia en los dos casos habiéndose producido distintas actividades de carrocería.

### 8.1.2 Escenario 2

En este escenario se estudiarán dos departamentos distintos de un mismo vehículo, en los cuales han ocurrido las mismas actividades de carrocería, pero en los que hay distintos COcupacion. Se muestra a continuación la tabla del escenario.

| **Departamento** | Copiloto | Trasero Izquierdo |
|---|---|---|
| **Ocupación** | ***Adulto*** | ***Carga*** |
| **Actividades** | 1. Abre puerta<br>2. Cierra puerta | 1. Abre puerta<br>2. Cierra puerta |

Tabla 3. Actividades del escenario de simulación 2.

Se podrá comprobar con esta simulación, si el sistema es capaz de inferir la ocupación correcta en cada departamento, habiendo ocurrido las mismas actividades. Lo cual demostraría que ha sido capaz de adaptarse a las validaciones del usuario "aprendiendo" qué ocupación hay en qué departamento.

## *8.2 Implementación de las simulaciones*

Se ha descrito en la sección 7.2.1 cómo el usuario puede introducir datos de itinerarios, tramos y ocupación a través del SGIA. Una vez generado el fichero de actividades sobre el que se quiere hacer inferencias, el usuario debe, a través del MMUC, ir introduciendo la ocupación real para cada tramo de ese itinerario. Es de esperar que, en cada ejecución, las inferencias del prototipo sean más acertadas. Sin embargo, el usuario debe seguir metiendo la ocupación real de cada tramo una y otra

vez para permitir que el MMUC vaya ajustando los pesos de las JPs. Esto hace muy tediosa y repetitiva la tarea de adaptación al usuario.

Como solución se desarrolló una nueva funcionalidad que permite realizar hasta 100 ejecuciones seguidas a partir de un fichero de actividades de carrocería y de un fichero de validación que contiene la ocupación real del habitáculo. Éste último fichero tiene una estructura XML y lo genera la aplicación cada vez que el usuario valida la ocupación de un tramo. Este fichero tiene el aspecto que muestra la Figura 35:

```
- <map>
    <departamento ubicacion="Conductor">Adulto</departamento>
    <departamento ubicacion="Copiloto">Adulto</departamento>
    <departamento ubicacion="TrasIzq">Bebe</departamento>
    <departamento ubicacion="TrasCentral">Vacio</departamento>
    <departamento ubicacion="TrasDcho">Niño</departamento>
    <departamento ubicacion="Maletero">Vacio</departamento>
  </map>
```

Figura 35. Estructura XML del fichero de validación.

Se añadió también en la pantalla inicial del prototipo una opción con el nuevo modo de ejecución, la cual permite seleccionar un fichero XML con las actividades que se quieren procesar, el fichero de validación con la ocupación real asociada a las actividades y el número de iteraciones que se quiere ejecutar la aplicación, tal y como se muestra en la Figura 36.

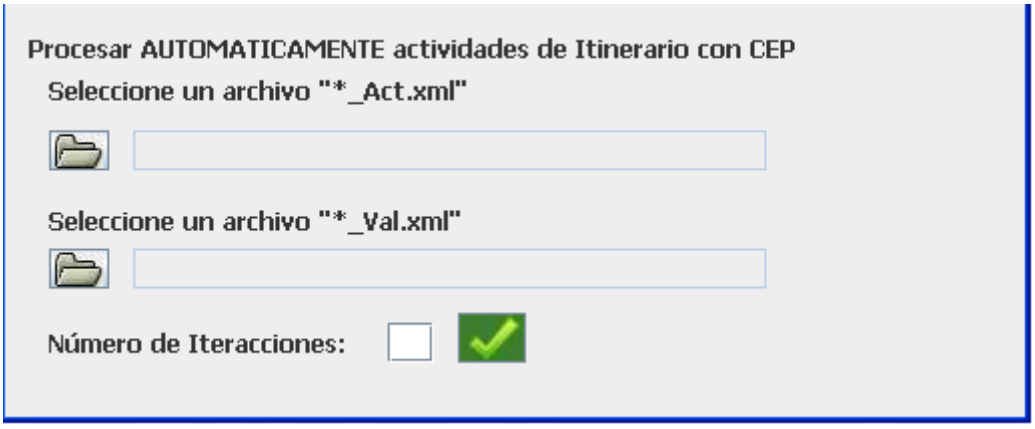



Figura 36. Opción para la ejecución automática de la aplicación

Si se elige este modo de ejecución, la aplicación sigue la misma línea de ejecución off-line definida en el presente documento con una única salvedad en el MMCU: en este caso cuando la ejecución del sistema llega a este módulo, en vez de mostrar al

usuario las inferencias de las ocupaciones de cada tramo esperando que él las valide o corrija, tomará del fichero de validación las ocupaciones reales, lanzando entonces el mecanismo de adaptación.

Además, cuando se detecta el final del itinerario, en lugar de finalizar el proceso, se volverán a introducir las actividades del mismo desde el principio iniciándose de este modo una nueva iteración. Así tantas veces como número de iteraciones haya indicado el usuario en el acceso.

Como resultado del proceso descrito se encontrará en el directorio *DATOS_RECOLECTADOS* de la raíz de la aplicación una carpeta por cada usuario donde se van almacenando los resultados de las distintas iteraciones. Por cada iteración se generan dos ficheros, uno en el que se almacenan las ocupaciones inferidas por el sistema y otro fichero de validación del habitáculo.

Para la generación de los dos escenarios descritos, se han creado dos usuarios en el sistema, usuario 1 y usuario 2. Mientras que el usuario 1 ha servido para simular el escenario 1 (estudiando el departamento trasero derecho de su habitáculo), el usuario 2 se ha utilizado tanto para el escenario 1 (estudiando el mismo departamento que para el usuario 1), como para el escenario 2 (estudiando los departamentos de copiloto y trasero izquierdo).

El usuario 1 ha introducido (a partir del modo de ejecución on-line) un itinerario de los que suele realizar, donde entre otras cosas, pone a un niño en el departamento trasero derecho del vehículo que es donde se va a centrar la atención para la simulación. Una vez que el usuario ha introducido el itinerario, el sistema realiza las inferencias para el habitáculo y se las muestra al usuario 1, el cual las valida con la ocupación real, lo cual genera el fichero de validación de la ocupación real. Se lanza de nuevo la aplicación haciendo uso de la nueva opción de ejecución, a partir de la cual se selecciona el fichero de actividades, el de validación de la ocupación y se especifica que se ejecute durante 100 iteraciones, obteniendo como resultado un fichero de inferencias por cada iteración cuyos resultados serán mostrados en las gráficas.

El usuario 2 ha introducido su propio itinerario, en el que encontramos, que al igual que el usuario 1 lleva un niño en el departamento trasero derecho, además, se observa que para los departamentos de copiloto y trasero izquierdo, se realizan las mismas acciones aunque la ocupación es distinta. Una vez que el usuario 2 ha introducido el itinerario, el resto de procesos hasta obtener los ficheros para la

creación de gráficas se han realizado operando del mismo modo que el descrito para el usuario 1.

## *8.3 Gráficas de resultados de las simulaciones*

Una vez realizadas todas las pruebas, se pasó a generar las gráficas para mostrar los resultados obtenidos. Dichas gráficas deben mostrar de alguna manera la "diferencia" entre la ocupación inferida por el sistema y la real.

Para ello hay que tener en cuenta que tanto los COcupacion inferidos por el sistema como los proporcionados por el usuario a través del fichero de validación pertenecen a una taxonomía con forma de árbol. Por tanto a la hora de determinar cómo de diferentes o parecidos son los COcupacion inferido y real, podría tenerse en cuenta como de lejano o cercano está el uno del otro dentro de las ramas de la taxonomía. Para ello se utilizó el concepto de Similitud Semántica-Jerárquica (SSJ) explicado en la sección 6.2.3.1.

### 8.3.1 Análisis de los resultados obtenidos

A continuación, se van a representar en gráficas cada uno de los escenarios propuestos al principio de la sección. El significado de los ejes de coordenadas de las mismas se describe a continuación:

- Eje X: Muestra el número de iteración así como el COcupacion recuperado del fichero de inferencias generado en la misma. Se ha tomado una muestra del conjunto total de ficheros de inferencia para la realización de las gráficas, concretamente se han ido leyendo ficheros de 10 en 10, suponiendo que el resultado de cada uno de ellos es representativo de un conjunto de ficheros a su alrededor.

- Eje Y: Representa el valor de la SSJ entre el COcupación inferido por el sistema y el COcupacion indicado por el usuario en el fichero de validación.

#### 8.3.1.1 Escenario 1

Este escenario pretende comparar las inferencias para un mismo COcupacion habiendo interaccionado el usuario de manera distinta con la carrocería.

En el departamento trasero derecho del habitáculo del vehículo del usuario 1 hay un niño y se han realizado las siguientes actividades:

1. Abre puerta
2. Pone cinturón seguridad
3. Cierra puerta
4. Movimiento de ventanilla
5. Bloqueo de ventanillas

La Figura 37 muestra la evolución del SSJ entre el COcupacion inferido por el sistema en cada iteración y el real, indicado por el usuario.

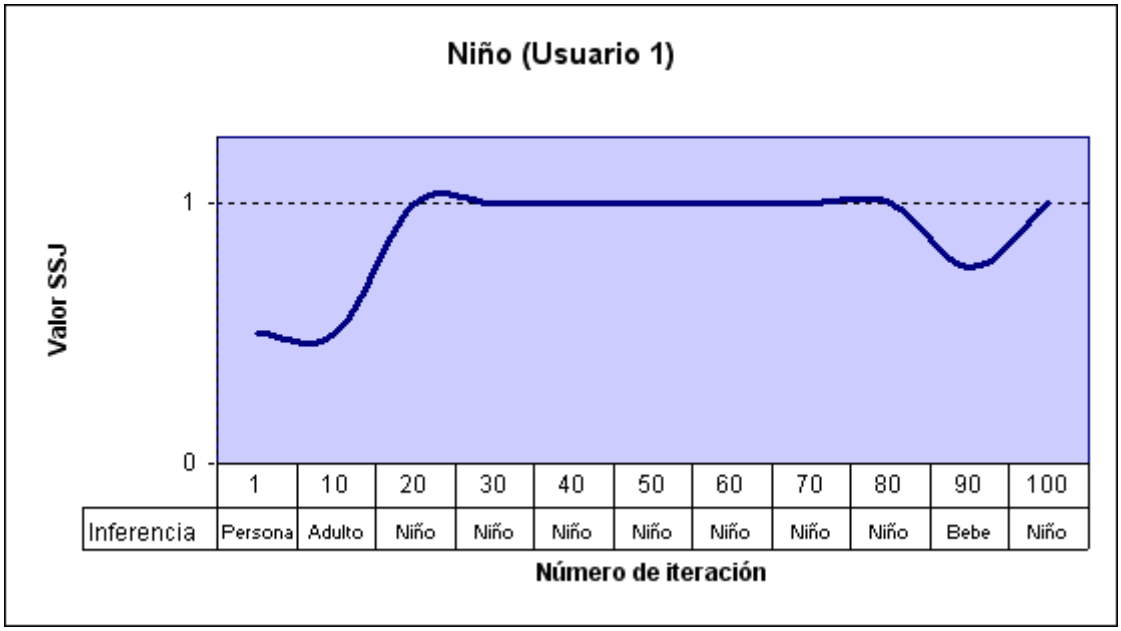



Figura 37. Gráfica del departamento trasero derecho del usuario 1 del escenario 1.

Se puede observar como el sistema ha ido variando las inferencias durante las primeras 10 iteraciones, a partir de las cuales comienza a tomar fuerza el COcupacion niño. En las primeras iteraciones el sistema se quedaba en un nivel de especificidad más bajo (persona y adulto tienen niveles de especificidad 2 y 3 respectivamente) y más tarde, cerca de la iteración 20, va aumentando la especificidad del COcupacion inferido (niño tiene nivel 4). Se observa que el peso del COcupacion niño en la JP de la regla correspondiente ha tomado un valor considerable, ya que el sistema casi siempre converge a este COcupacion. En la iteración 90, se ve como el sistema se ha desviado hacía el COcupacion hermano de niño dentro de su nivel de especificidad, esto es por que el mecanismo de selección utiliza valores aleatorios para elegir el COcupacion inferido en función de su probabilidad, la cual se va adaptando.

Se pasa a estudiar los resultados obtenidos para el departamento trasero derecho del habitáculo del vehículo del usuario 2 donde hay un niño y sobre el que se han detectado las siguientes actividades:

1. Abre puerta
2. Pone cinturón seguridad
3. Cierra puerta

Se muestra la evolución del SSJ entre el COcupacion inferido por el sistema en cada iteración y el indicado por el usuario en la Figura 38.

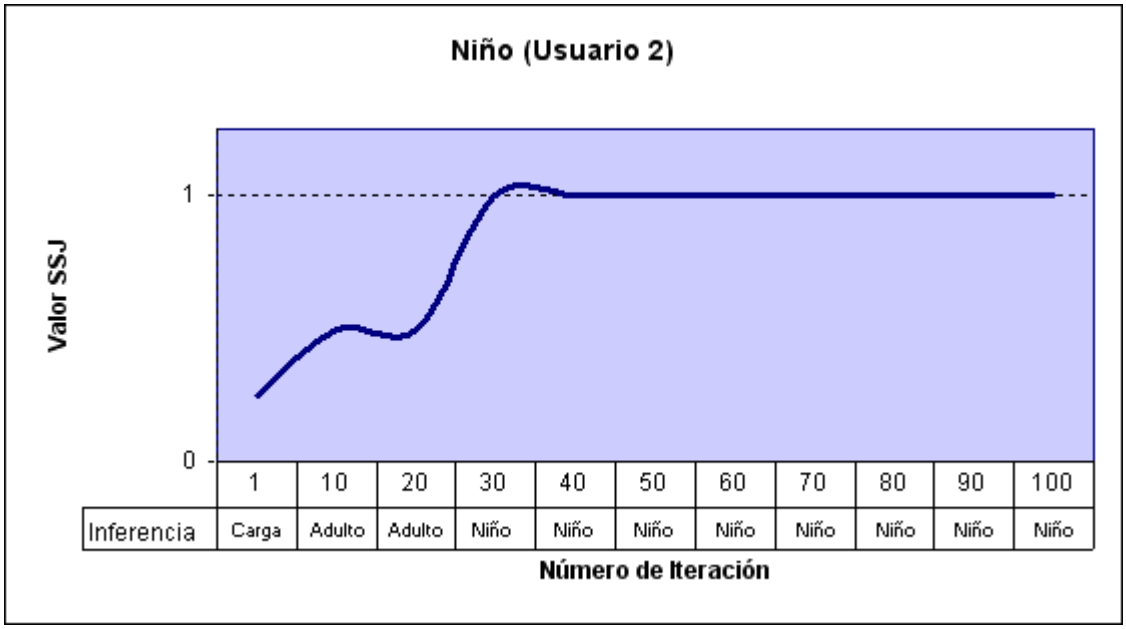



Figura 38. Gráfica del departamento trasero derecho del usuario 2 del escenario 1.

En este caso, al haberse realizado menos actividades de carrocería, al sistema le ha costado un poco más converger al COcupacion correcto, ya que, a partir los eventos que generan las actividades mencionadas no se llega a los EPAs de nivel 3 y 4, que son los que están configurados inicialmente (por la configuración inicial del perfil de usuario) para inferir COcupacion correspondiente a sus niveles. Por lo tanto, serán los EPAs de nivel 1 o 2 los que tengan que inferir el COcupacion niño, el cual, por la configuración inicial tiene un peso con valor 0 en las JPs asociadas a las EPRs de estos EPAs. A partir de la iteración 30 vemos que el peso del COcupacion niño ha tomado un valor importante, ya que a partir de este punto las inferencias convergen a el. Lo que demuestra que el sistema ha sido capaz de adaptarse al usuario "aprendiendo" que en el departamento trasero derecho hay un niño, aunque las actividades no lo evidenciasen.

Para el usuario 1 el sistema ha convergido más rápido al valor del COcupacion correcto ya que las actividades que se producen aportan más información al sistema que las generadas por el usuario 2. Aún así, el sistema se ha adaptado también al

comportamiento el usuario 2 sin haber una diferencia muy grande en número de iteraciones necesarias para la convergencia al COcupacion correcto.

### 8.3.1.2 Escenario 2

En este escenario se pretende comprobar si el sistema es capaz de adaptarse a distintos COcupacion a partir de las mismas actividades de carrocería producidas en diferentes departamentos del habitáculo de un mismo vehículo. Concretamente, se van a estudiar los departamentos de copiloto y trasero izquierdo del habitáculo de un mismo vehículo.

El usuario del vehículo en este segundo escenario, suele dejar una caga atrás e ir con un acompañante en el departamento del copiloto que no se suele poner nunca el cinturón. Con lo cual, el sistema, para estos dos departamentos escucha las mismas actividades, que son:

1. Abre puerta
2. Cierra puerta

Las Figura 39 y 40 muestran las dos gráficas generadas en el proceso de simulación del itinerario de este usuario:

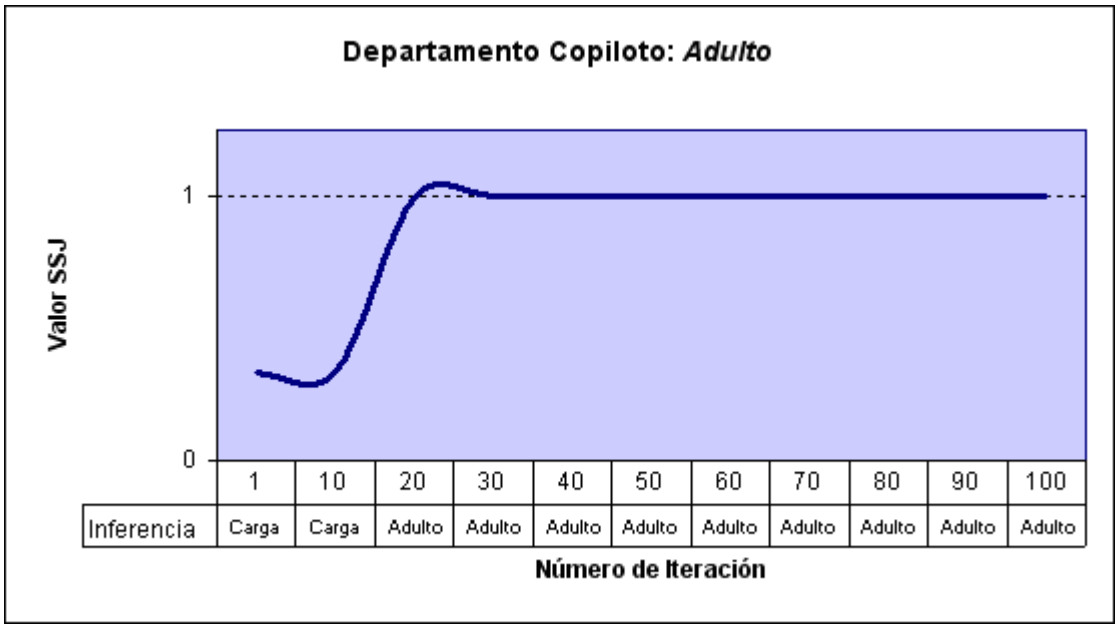



Figura 39. Gráfica del departamento copiloto con un adulto. Escenario 2.

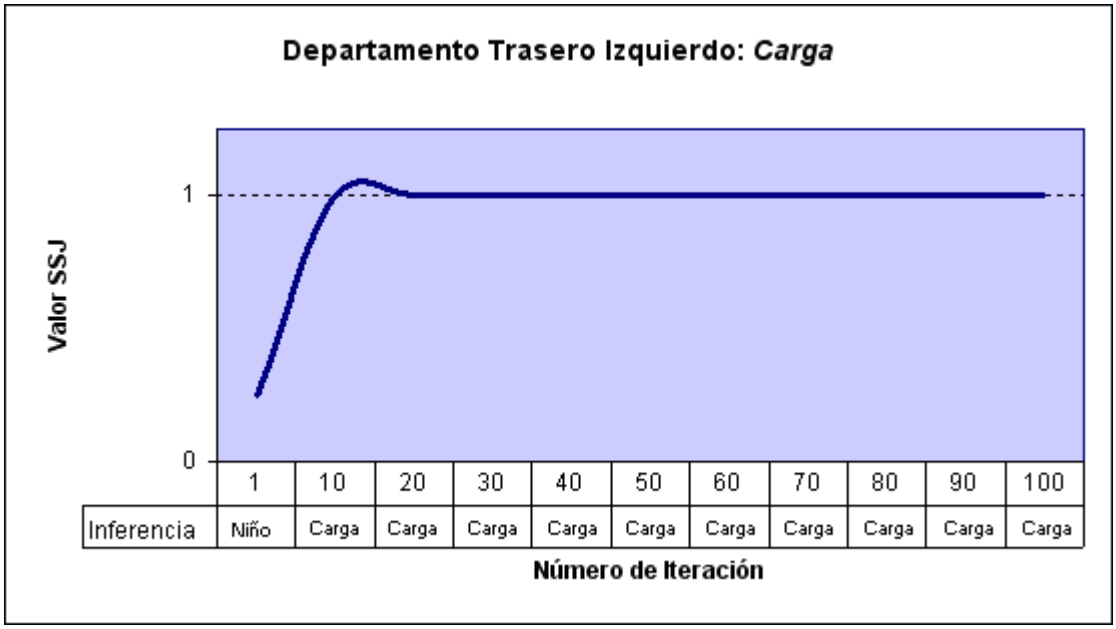



Figura 40. Gráfica del departamento copiloto con un adulto. Escenario 2.

El sistema está configurado por defecto para que, ante los eventos de abrir y cerrar puerta, si no se detecta ningún síntoma de persona, se infiera el COcupacion carga. Vemos en la gráfica de la Figura 40 que ya en las primeras 10 iteraciones el sistema converge a carga teniendo este COcupacion una probabilidad muy alta de salir. Sin embargo la gráfica de la Figura 39 tarda un poco más en converger. Esto es debido, como ya se ha explicado anteriormente, a que con los eventos generados no se llega a los EPAs de nivel 3, los cuales tienen más probabilidades de inferir adulto. Por lo tanto, el sistema necesita más iteraciones para que el COcupacion adulto tome más peso en EPAs de nivel inferior. Como se puede ver en la Figura 40, esto ocurre a partir de la iteración 30.

Si se comparan los dos escenarios se observa que el sistema, independientemente de las actividades de carrocería generadas, suele converger a un COcupacion correcto como mucho a partir de las 30 iteraciones, a partir de las cuales la probabilidad de que se infiera un COcupacion erróneo es muy pequeña.

# 9 Conclusiones y vías futuras

## *9.1 Conclusiones*

En el presente trabajo se ha desarrollado un sistema context-aware que, a partir de los sensores de un vehículo, genera información sobre el contexto interno del mismo. La herramienta permite recolectar, etiquetar, almacenar y procesar datos procedentes de un simulador de actividades de la carrocería de un vehículo de gama media, en concreto de un *Citroën C4 Collection.* Para la consecución de la herramienta con sus funcionalidades se han tenido que realizar diversas tareas que se describen a continuación.

Se ha realizado un estado del arte sobre los sensores más importantes que hay en el mercado de vehículos, centrando la atención en los referentes a la carrocería, realizando un análisis sobre cuales de estos datos podrían ser útiles en el sistema para proporcionar la información.

La necesidad de leer los datos que proporcionan los sensores de los vehículos hizo que la línea de trabajo se centrara en el estudio del protocolo OBD II que es un estándar implementado por ley en todos los vehículos a partir de 2003. Este protocolo, entre otras cosas, tiene una codificación específica capaz de almacenar información sobre los datos de los sensores que existen en un vehículo. Esto ha llevado a descubrir que estos códigos de carrocería son propietarios de los fabricantes de vehículos y su obtención supone un desembolso de dinero. No obstante, como conclusión destacamos que se ha confirmado la viabilidad de la implantación de nuestro sistema en un vehículo real.

Por otro lado, se ha profundizado en el conocimiento de la disciplina de Procesamiento de Eventos Complejos (CEP, Complex Event Processing) como mecanismo de diseño de sistemas capaces de detectar situaciones de interés, en este caso todo lo concerniente al contexto interno de un vehículo. Concretamente la API Esper y todas sus funcionalidades es lo que se ha usado la el desarrollo de CEP en nuestro sistema.

Además de todos los logros obtenidos durante el proceso de desarrollo del sistema, otra aportación a un algoritmo, en las líneas del aprendizaje por refuerzo, para adaptar el perfil del usuario almacenado a medida que este interacciona con el sistema. Para el desarrollo de este algoritmo se ha diseñado una taxonomía de ocupación del habitáculo del vehículo, la cual contempla un orden entre sus elementos

(atendiendo a los conceptos de especificidad y fragilidad). Esta taxonomía es la base para la definición de las recompensas/penalizaciones que guían el proceso de adaptación.

Como conclusión final hay que destacar que en este proyecto se conectan varias disciplinas como son Procesamiento de Eventos Complejos, Aprendizaje Computacional, Context-Awareness, para desarrollar un sistema que pertenece al ámbito de los Sistemas Inteligentes de Transporte.

## *9.2 Trabajos futuros*

Se pueden distinguir distintos tipos de líneas de trabajo para dar continuidad al presente proyecto. Se pasa a continuación a describir cada una de ellas:

- **Posibles extensiones y mejoras del prototipo implementado**

  - Incorporación de varios tipos de filtros de sensorización. De esta manera se modificaría el simulador de actividades de carrocería para que el usuario seleccionara un filtro de sensorización en función de el tipo de vehículo que desee simular (alta gama, gama media, etc.).

  - Implementación del protocolo OBD-II. Con el objetivo de tomar datos reales de un vehículo, se necesita sustituir el simulador ya mencionado por un adaptador OBD II. De esta manera el sistema sería capaz de recoger los datos desde del CAN BUS. Debido a la problemática de los códigos de carrocería propietarios expuesta, esta vía de trabajo futuro está sujeta a la disponibilidad de la cantidad de dinero necesaria para poder acceder a dichos códigos.

  - Modificación de los mapas. Permitir al usuario cargar en el simulador el mapa que le interese atendiendo a los itinerarios que suela realizar, para ello se puede pensar en incorporar alguna herramienta existente en el mercado para la gestión de mapas.

  - Consolidación de los tramos e itinerarios inferidos. El MMUC tiene un mecanismo de consolidación de las inferencias de la ocupación de los tramos. No obstante, el MGAI también hace inferencias sobre itinerarios y los distintos tramos que lo componen que serán almacenadas en un fichero XML, pero no se muestran al usuario para que pueda validar su exactitud. Una extensión de la aplicación podría ser el desarrollo de un mecanismo que permita al usuario

hacer correcciones sobre la forma en la que el sistema ha detectado los tramos de un itinerario.

- Conectar el sistema con una VANET. Esta implementación serviría para, por ejemplo, enviar información sobre la ocupación del habitáculo a un sistema central de tráfico para que se pudiera aumentar la percepción de la situación interna de un vehículo por parte de las entidades encargadas de controlar el tráfico rodado en carreteras y autovías.

- **Profundizar y mejorar el algoritmo de adaptación**

  - Una de las tareas de investigación inmediatas será la de ahondar más profundamente en el mecanismo de adaptación aquí propuesto, el cual debe considerarse solo como una primera aproximación.

    a) Realizar muchas más pruebas. Para el presente trabajo, se han realizados varias pruebas de las que se ha hecho un análisis de resultados. Pero se es consciente de que para una sustentación más fiable del algoritmo serían necesarias cientos o miles de pruebas extra.

    b) Modificar las funciones de penalización y recompensa para:

       i. Tener en cuenta tanto el valor SSJ como la fragilidad asumiendo que es más penalizable que se infiera carga cuando es persona, que el hecho de inferir persona cuando es carga, ya que una ocupación persona ha sido considerada más frágil que una ocupación carga.

       ii. Evitar la necesidad de supervisión por parte del usuario. De modo que los EPAs de cierto nivel obtengan refuerzo/penalización desde los EPAs de niveles superiores. En este sentido el algoritmo se acercaría más a la idea de aprendizaje por refuerzo. Por ejemplo, si un EPA de nivel 4 infiriera bebé esto podría servir para penalizar a los EPAs de niveles inferiores que hicieron una inferencia incoherente.

    c) Adaptar, no solo la salida que producen los patrones CEP, sino también los mismos patrones y reglas para cada usuario en particular.

d) Manejar varios perfiles para el mismo usuario distinguiendo entre los diferentes itinerarios que éste puede efectuar, ya que por ejemplo, hay usuarios que no se ponen el cinturón por ciudad pero sí por autovía, con lo que el usuario debería tener un perfil para ciudad y otro distinto cuando el itinerario discurra por una autovía.

- **Otras posibles líneas de trabajo**

  - Una posible línea de trabajo que podría dar continuidad al presente proyecto sería la expuesta en la sección 1, que trata de contextualizar tramos con la consecuente necesidad de utilizar ontologías, GPS real y GIS (Sistema de Información Geográfica, Geographic Information System) así como incorporar mecanismos de predicción del itinerario. Por ejemplo, que el sistema sea capaz de saber que, si se coge el coche sobre las 9 de mañana y se monta un niño, el usuario va al colegio, más tarde, cuando se baja el niño, el sistema sabrá que a continuación el usuario se va al trabajo. Este mecanismo de predicción permitiría la recepción de información acorde con el contexto interno del vehículo y sugeriría servicios, como compartición de plazas de aparcamiento, etc.

  - Aplicar mecanismos de aprendizaje off-line a los datos recolectados. Para ello, sería necesario que un número suficientemente grande de usuarios introdujera sus itinerarios, generándose así gran cantidad de datos sobre los que poder, entre otras cosas, buscar patrones de comportamiento.

Como se ha expuesto se ha abierto un camino muy largo e interesante de trabajo futuro.

# 10 Referencias

[1] F. T. Durso, A. Sethumadhavan, *Situation awareness: Understanding dynamic environments,* Human Factors: The Journal of the Human Factors and Ergonomics Society, 50(3):442–448, 2008.

[2] Dirección General de Tráfico, estadísticas e indicadores. http://www.dgt.es.

[3] Noticia dispositivo alerta olvidos, http://www.20minutos.es/noticia/500740/3.

[4] United States Patent. Patent No. US 7,319,382 B1, Date of Patent Jan. 15, 2008.

[5] W. Barfield, T. A. Dingus, *Human factors in intelligent transportation systems*, Lawrence Erlbaum Associates Inc. Publishers, 1998.

[6] M. A. Chowdhury, A. W. Sadek, *Fundamentals of intelligent transportation systems planning,* Artech House Inc., 2003.

[7] F. Terroso-Saenz, M. Valdés-Vela, *Fuzzy Modeling for Vehicle Maneuver Detection in a Scene*, Fuzz-IEEE, 2010.

[8] R. Toledo-Moreo, M. Pinzolas-Prado, J. M. Cano-Izquierdo, *Maneuver prediction for road vehicles base don a neuro-fuzzy architecture with a low cost navigation unit*, IEEE Transactions on Intelligent Transportation Systems, 2009.

[9] J. Santa Lozano, *Arquitectura de una plataforma telemática integral para el despliegue de servicios Ubicuos en el ámbito de los sistemas inteligentes de transporte*, Universidad de Murcia, 2009.

[10] J. Dunkel, A. Fernández, R. Ortiz, S. Ossowski, *Event-Driven Architecture for decision Support in Traffic Management Systems*, Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems, 2008.

[11] C. Sotomayor, F. Terroso, M. Valdés, A. F Gómez, *Arquitectura dirigida por eventos para la gestión de la información generada por redes vehiculares*, X Congreso Español ITS, 2010.

[12] F. Dodino, *XML*, Universidad Tecnológica Nacional, Septiembre 2008.

[13] Extensible Markup Language (XML), http://www.w3.org/XML.

[14] D. Luckham, R. Schulte, *Event Processing Glossary - Version 1.1*, 2008.

[15] O. Pawlowski, J. Dunkel, R. Bruns, S. Ossowski, *Applying Event Stream Processing on Traffic Problem Detection,* 14th Portuguese Conference on Artificial Intelligence, EPIA, 2009.

[16] B. Morillas Bueno, E. Irigoyen Gordo, *Sensores en el automóvil*, Universidad del País Vasco, Escuela Superior de Ingenieros.

[17] M. Concepción, *Estrategias de sistemas OBD-II,* 2010.

[18] OBD II Trouble Codes Home. http://www.obd-codes.com.

[19] Workshop Information System 3.0 (WIS), Saab Automobile AB, 2005.

[20] Equipment and Tools Institute, http://www.etools.org/.

[21] R. Rada, H. Mili, E. Bicknell, M. Blettner, *Development and application of a metric on semantic nets*, IEEE Transactions on Systems, Man, and Cybernetics, 19(1):17.30, 1989.

[22] A. Arasu, S. Babu, J. Widom, *The CQL Continuous Query Language: Semantic Foundations and Query Execution*, 9th International Conference on Data Base Programming Languages (DBPL), pp 1-19, 2003.

[23] L. Pack Kaelbling, M. L. Littman, A. W. Moore, *Rein-forcement Learning: a survey*, AI access fundation and Morgan Kaufmann Publishers, 1996.

[24] J. S. R. Jang, C. T. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing.A computational Approach to Learning and Machine Intelligence*, Prentice-Hall Inc., 1997.

[25] S. E. Dreyfus, A. M. Law, *The art and theory of dynamic programming*, volume 130 of Mathematics in Science and Engineering, Academic Press, 1977.

[26] NetBeans IDE 6.8. http://netbeans.org.

[27] XStream Proyect. XStrem Reference Documentation, version 1.3.1, http://xstream.codehaus.org, 2010.

[28] Espertech, Esper Reference Documentation, version 3.4.0, http://esper.codehouse.org, 2010.