# Caso de estudio del proyecto de Grado

El sistema de parqueo de la Pontifica Universidad Javeriana Cali (PUJC) ha sido seleccionado como el caso de estudio para realizar en conjunto las verificaciones formales y pruebas planteadas en el presente Trabajo de Grado, con el fin de valorar y minimizar los errores en el diseño de un sistema distribuido.

La figura 1 muestra el sistema de parqueo de vehículos de la PUJC, el cual cuenta con las siguientes características:

1. Por lo menos con una entrada y una salida principal, para este caso se encuentran ubicadas sobre la avenida Cañasgordas.
2. Cada entrada y salida principal cuenta con un sistema que permite identificar las placas de los carros, un lector de carnés y un sistema básico de sensores que garantiza que el usuario pueda ingresar o salir del sistema de parqueo.
3. Un conjunto de zonas[[1]](#footnote-1) que están bajo el mando de un controlador; en la figura 1 se muestran 5 controladores del sistema de parqueo de la PUJC que son: Principal, Las Palmas, El Lago, Docentes y Almendros.
4. Repositorios donde se encuentran los usuarios que tienen permitido el acceso al sistema de parqueo y sus placas registradas.

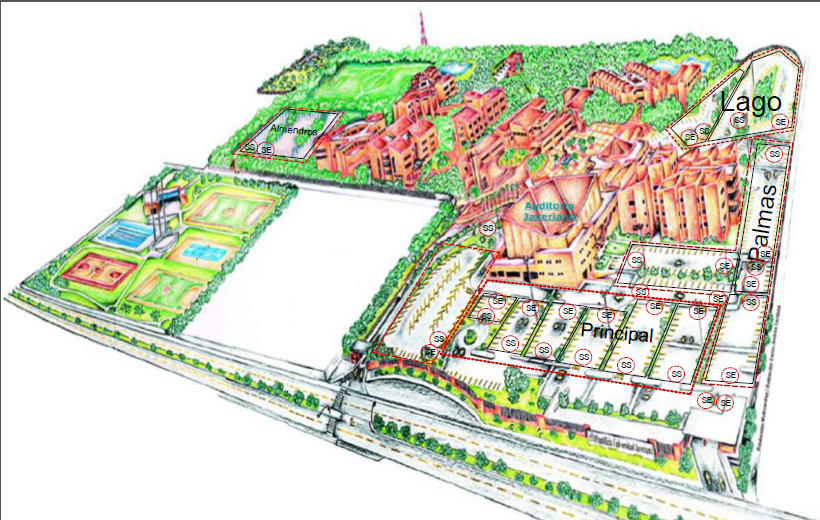


Figura 1. Sistema de Parqueo de la PUJC

La figura 1 muestra básicamente 5 controladores de zonas que son:

1. Controlador “Principal” que tiene 6 zonas de parqueo.
2. Controlador “Docentes” que tiene 1 zona de parqueo.
3. Controlador “Las Palmas” que tiene 3 zonas de parqueo.
4. Controlador “El Lago” que tiene 2 zonas de parqueo.
5. Controlador “Los Almendros” que tiene 1 zona de parqueos.

La figura 2 ilustra los componentes del sistema de parqueo de la PUJC. Básicamente un sistema de parqueo de vehículos cuenta con *C* controladores de zonas y a su vez puede tener hasta *Z* zonas; éstas últimas poseen un sistema de sensores capaces de informar a su controlador si un vehículo ingresó o salió de dicha zona. Adicionalmente, el sistema puede tener hasta *M* entradas y *N* salidas principales.

Para garantizar que al sistema sólo ingresen usuarios registrados como (estudiantes, docentes, personal administrativo, colaboradores, etc.), es indispensable contar con elementos que permitan cumplir dicho objetivo; por ello, el sistema está diseñado para que cada entrada y salida principal cuente con una cámara, cuyo propósito es detectar la placa del vehículo que va a ingresar o salir del sistema. Igualmente, se considera tener un lector de carnés que permite validar si el usuario efectivamente hace parte del sistema; finalmente se requiere un conjunto de sensores que permitan que el vehículo ingrese o salga del sistema sin ocasionar daño alguno al vehículo.

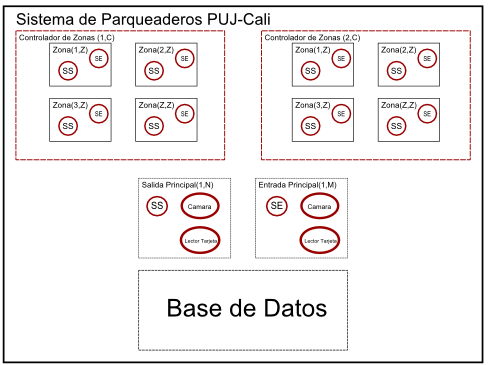


Figura 2. Componentes del Sistema de Parqueo de la PUJC

La tabla 1 describe las funciones que posee el sistema de parqueo de la PUJC y el tipo de usuario que puede hacer uso de éstas.

|  |  |  |
| --- | --- | --- |
| **Función** | **Descripción de la función** | **Usuario** |
| Crear Controlador de Zona | Permite que el sistema tenga más controladores y a su vez más capacidad de zonas que se traduce en aparcar más vehículos. | ADMINISTRADOR |
| Crear una Zona en un Controlador Específico | Permite anexar una zona en un controlador específico ampliando la capacidad de éste último. | ADMINISTRADOR |
| Crear una Entrada o Salida principal | Permite que el sistema tenga más entradas y salidas para el acceso de usuarios al sistema de parqueo. | ADMINISTRADOR |
| Solicitar información a los controladores de zona | Permite saber las plazas libres que posee cada uno de los controladores, lo que posibilita que el usuario sepa en qué controlador y en qué zona hay espacio para aparcar. | ADMINISTRADOR/AUTOGESTIÓN DEL SISTEMA[[2]](#footnote-2) |
| Inicializar las plazas libres y totales de una zona de un controlador en específico | Permite inicializar las plazas totales y libres de una zona de un controlador en específico, posibilita tener zonas de diferentes capacidades. | ADMINISTRADOR |
| Reportar ingreso o salida de un usuario al sistema de parqueo | Reporta si un usuario ha ingresado al sistema de parqueo si y solo si: El código del usuario y la placa de su vehículo se encuentra registrado en los repositorios de datos del sistema de parqueo. | USUARIO |
| Reportar el ingreso o salida de un usuario en una zona | Reporta si un vehículo ha ingresado o ha dejado una zona cuando se detecte una secuencia específica en los sensores de la zona. | USUARIO |

Tabla 1 Funciones del sistema de parqueo de la PUJC

## Modelado

Para realizar el modelado del sistema de parqueo de la PUJC, se hizo uso del lenguaje SDL sobre la herramienta RTDS; el modelo del sistema cuenta con las funciones descritas en la tabla 1.

La arquitectura del sistema se divide en dos grandes bloques y estos a su vez se dividen en más bloques y procesos. La figura 3 detalla la arquitectura general del sistema de parqueo de la PUJC.

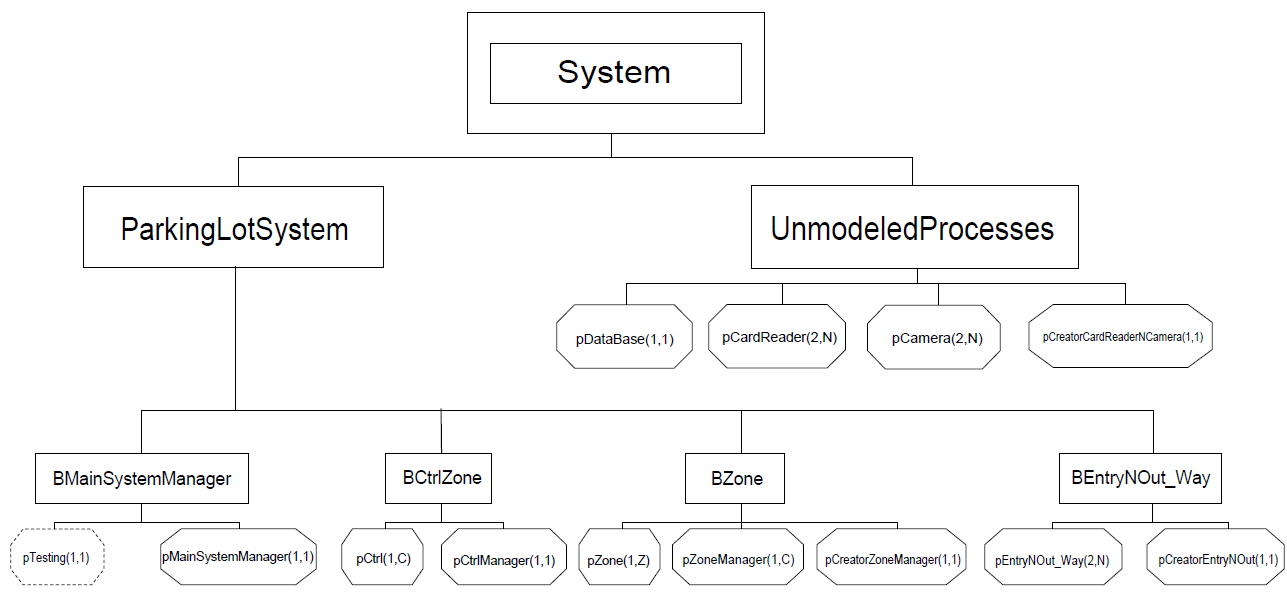


Figura 3. Arquitectura del modelo del sistema de parqueo de la PUJC

Como se aprecia en la figura 3, el sistema cuenta con dos bloques principales llamados ParkingLotSystem y UnmodeledProcesses. El primer bloque está conformado por los procesos y elementos que se han modelado con un mayor nivel de detalle; por el contrario, en el segundo bloque se encuentran los elementos del sistema que no se han modelado sino que se han considerado como procesos, los cuales reciben un valor y retornan otro pero sin realizarse un procesamiento específico de la información, dado que no es de interés del proyecto modelarlos.

Las tablas 2 y 3 describen las funciones generales de cada bloque y de los procesos del sistema de parqueo, respectivamente:

|  |  |
| --- | --- |
| **Nombre del bloque** | **Propósito/Función** |
| ParkingLotSystem | Tiene como propósito agrupar los bloques BMainSystemManager, BCtrlZone y BZone. Adicionalmente es el bloque que tiene más interacción con el Administrador. |
| UnmodeledProcesses | Tiene como propósito agrupar los procesos no modelados que son: pDataBase, pCardReader, pCamera y pCreatorCardReaderNCamera. En este bloque se da lugar a la validación de usuarios permitidos para ingresar o salir del sistema de parqueo. |
| BMainSystemManager | Agrupa los procesos pMainSystemManager y el proceso pTesting[[3]](#footnote-3), este bloque interactúa principalmente con el Administrador. |
| BCtrlZone | Agrupa los procesos pCtrl y pCtrlManager, interactúa básicamente con los bloques BMainSystemManager y BZone. |
| BZone | Agrupa los procesos pZone, pZoneManager, pCreatorZoneManager. Interactúa principalmente con el bloque BCtrlZone y el usuario. |
| BEntryNOut\_Way | Agrupa los procesos pEntryNOut\_Way y pCreatorEntryNOut\_Way. Interactúa principalmente con los procesos del bloque UnmodeledProcesses y con el usuario. |

Tabla 2. Descripción de los bloques del sistema de parqueo.

|  |  |
| --- | --- |
| **Nombre del proceso** | **Funciones** |
| pDataBase | Este proceso representa los repositorios de datos del sistema, además está encargado de evaluar si un usuario está habilitado para ingresar o salir del sistema de parqueo. |
| pCardReader | Este proceso representa al lector de carnés y se encarga de entregar el código del usuario que está por ingresa o salir del sistema de parqueo. |
| pCamera | Este proceso representa la cámara que captura la placa del vehículo y la entrega al sistema para evaluar si el usuario está permitido para ingresar o salir del sistema. |
| pCreatorCardReaderNCamera | Proceso encargado de crear los procesos pCardReader y pCamera, y asignarlos a su respectivo proceso pEntryNOut\_Way. |
| pEntryNOut\_Way | Este proceso representa una entrada o salida principal, es el encargado de interactuar con el usuario. Adicionalmente es el único proceso que interactúa con los procesos pDataBase, pCardReader, pCamera y pCreatorCardReaderNCamera. |
| pCreatorEntryNOut\_Way | Proceso encargado de crear el proceso pEntryNOut\_Way. |
| PZone | Este proceso representa una zona del sistema e interactúa con el controlador de zonas reportando el ingreso o salida de un vehículo, dependiendo de la secuencia de señales del conjunto de sensores que se reciba. |
| pZoneManager | Proceso encargado de crear instancias del proceso pZone y a su vez asignarle su respectivo proceso pCtrl. |
| pCreatorZoneManager | Proceso encargado de crear los procesos pZoneManager y asignarle a éste su respectivo controlador de zonas. |
| pCtrl | Este proceso representa un controlador de zonas y está encargado de controlar los procesos pZone, adicionalmente reporta el estado de las plazas libres y totales que posee cada zona. |
| pCtrlManager | Proceso encargado de crear los procesos pCtrl. |
| pMainSystemManager | Este proceso representa la interfaz directa entre el administrador y el sistema, a través de este proceso se puede crear entradas y salidas principales, controladores de zona, zonas, hacer requerimientos de información, inicializar tanto plazas libres como plazas totales. |
| pTesting | Proceso que permite acoplar el sistema para hacer pruebas funcionales de caja negra. Este proceso no debe ser considerado en la fase de implementación del sistema de parqueo. |

Tabla 3. Descripción los procesos del sistema de parqueo.

Dentro de la arquitectura del sistema de parqueo presentada en la figura 3 se detallan algunos procesos que tienen como función crear instancias, para más información de dichos procesos ver la tabla 3. Dado que la arquitectura del sistema de parqueo se compone de bloques, no es posible crear instancias de procesos si éstos no se encuentran en el mismo nivel, lo anterior justifica la existencia de dichos procesos como *puentes* para la creación de procesos que se encuentren en su mismo nivel.

### Especificación del sistema de parqueo

El modelado del sistema de parqueo empieza desde los requerimientos de especificación del sistema, en este caso se desea que éste posea las funciones descritas en la tabla 1. La herramienta RTDS tiene la posibilidad de anexar al proyecto diagramas MSC, éstos son fundamentales para describir las especificaciones del sistema. Existen estrategias para el diseño y desarrollo de modelos, entre las cuales encontramos Top-Down y Bottom-Up. Top-Down es una estrategia donde se empieza desarrollando el modelo desde un alto nivel con una mínima abstracción y termina en bloques o procesos con altos niveles de abstracción. Bottom-Up es una estrategia que consiste en empezar a diseñar los elementos del sistema que tiene un alto nivel de abstracción y a partir de éstos se crean bloques más complejos. En el modelado del sistema de parqueo de la PUJC se hace uso de la estrategia Top-Down para la especificación de requerimientos del sistema que se encuentran descritos en diagramas de MSC, y para la implementación del sistema se hace uso de la metodología Botton-Up empleando SDL.

#### MSCs de las especificaciones del sistema

En la figura 3 detalla que el sistema de parqueo posee dos grandes bloques y a su vez estos contienen sub-bloques y procesos. Dado que la practicidad de especificar el sistema en alto nivel sin tener en cuenta un alto nivel de abstracción, es mucho más sencillo que hacer el proceso contrario; se elabora un MSC que tiene como finalidad mostrar un escenario en el cual se describa la interacción de los dos bloques principales: ParkingLotSystem y UnmodeledProcesses. La figura 4 muestra la interacción entre los dos bloques ParkingLotSystem y UnmodeledProcesses, el escenario que se describe es el acceso de un vehículo al sistema de parqueo. En este MSC encontramos tres agentes interactuando entre sí los cuales son: Env, ParkingLotSystem y UnmodeledProcesses. El agente Env representa el ambiente o medio por el cual el sistema intercambia mensajes con el usuario o con los sensores. El procedimiento que describe el acceso de un vehículo al sistema de parqueo de la PUJC está dado por la siguiente secuencia:

1. El proceso Env envía una señal al bloque ParkingLotSystem llamada sLoopInductive\_Entrance que indica que un vehículo está en la portería principal.
2. El bloque ParkingLotSystem necesita verificar que el usuario que desea ingresar al sistema de parqueo está autorizado. Para ello, necesita saber su identificación y la placa del vehículo. Con la señal sEnableCarReader habilita el lector de carnés.
3. Una vez que el bloque ParkingLotSystem tenga el código del usuario, habilita la toma de la foto a la placa, para ello intercambia el mensaje sRequestPlate con el bloque UnmodeledProcesses.
4. El bloque UnmodeledProcesses envía la placa como un charstring al bloque que hizo la consulta.
5. Una vez que se tenga la información del usuario es indispensable validarla y determinar si se le otorga el acceso o no al sistema. Para ello envía un requerimiento al bloque UnmodeledProcesses esperando la señal sOkUser que confirma que el usuario está habilitado para usar el sistema de parqueo o sNoRegis\_User que indica que el usuario no está permitido para usar el sistema de parqueo.
6. Si el usuario está habilitado para usar el sistema de parqueo entonces se efectúa una serie de pasos para darle bienvenida al usuario. Para ello se envía una señal a la talanquera para que permita el paso del vehículo, una vez que éste haya pasado, un sensor envía dicha señal indicando que es seguro bajar la talanquera y el bloque ParkingLotSystem culmina enviando la señal sDownBarrier, quedando así disponibles los dos bloques para interactuar nuevamente.

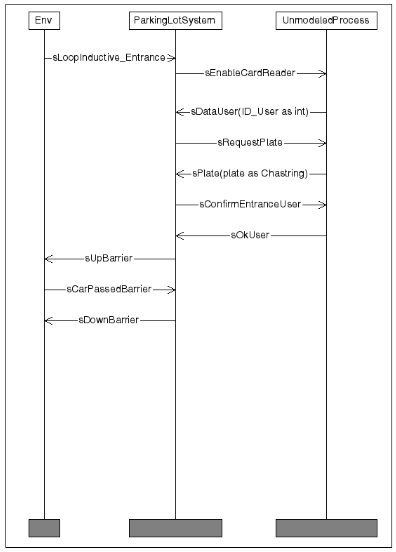


Figura 4. MSC ingreso vehículo al sistema de parqueo de la PUJC

El escenario que representa la salida de un vehículo del sistema de parqueo de la PUJC es similar al comportamiento descrito en la figura 4, la diferencia radica que el agente Env o entorno envía una señal sLoopInductive\_Exit y en el momento que el bloque ParkingLotSystem desea verificar si el usuario está autorizado para salir, envía una señal sConfirmOutUser.

Los MSCs son útiles para describir la interacción entre agentes, que si bien pueden ser entre bloques también sirven para describir el comportamiento a nivel de procesos. Como se muestran en las figuras 5 y 6 que representan los MSCs de la inicialización del sistema cuando éste empieza por primera vez. Los requerimientos del sistema son los siguientes:

* El sistema debe de tener por lo menos una entrada y una salida principal.
* Cada entrada y salida principal debe una cámara la cual retorna el valor de la placa del vehículo a partir de una foto y un lector de carnés.
* El sistema tiene por lo menos una zona con su respectivo controlador. La zona cuenta con sus respectivos sensores para indicar si un vehículo ha ingresado o salido de dicha zona.
* Cuenta con un repositorio de datos donde se encuentran los códigos y sus placas asociadas que están autorizadas para el acceso o salida del sistema de parqueo de la PUJC.

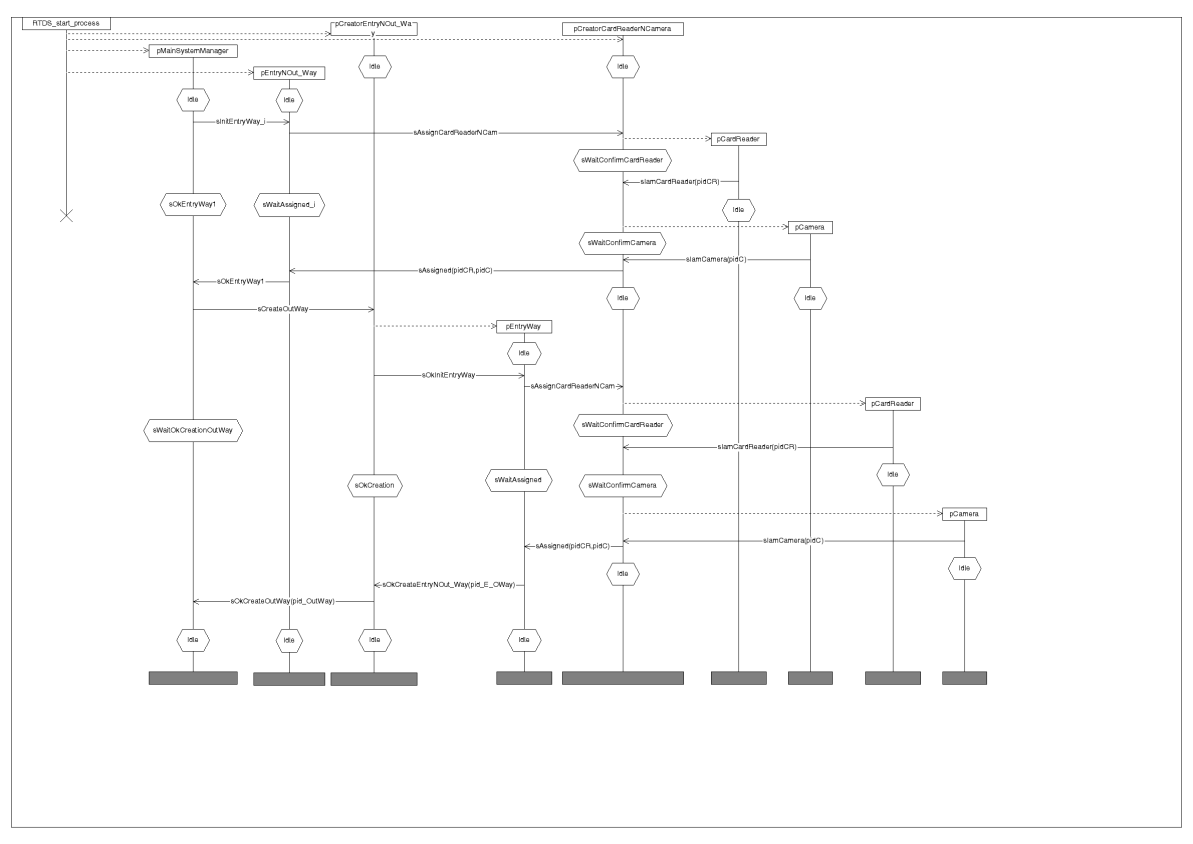


Figura 5. Inicialización entrada y salida principal del sistema

La figura 5, detalla la interacción entre procesos del bloque ParkingLotSystem y los procesos del bloque UnmolededProcesses. Inicialmente el sistema contiene una instancia de cada uno de los procesos de la figura 3. El proceso pMainSystemManager se encarga de asociar el proceso pEntryNOut\_Way a su tabla de entradas principales; para ello envía una señal llamada sInitEntryWay\_i al proceso pEntryNOut\_Way. A partir de esta solicitud lo que va a pasar es que éste proceso necesita que se le sea asignado un lector de carnés, pCardReader, y una cámara, pCamera. El proceso que tiene la facultad de instanciar estos procesos es pCreatorCardReaderNCamera, una vez éste haya creado el lector de carnés y la cámara, enviará una señal al proceso pEntryNOut\_Way con los identificadores correspondientes. El proceso pMainSystemManager cuando reciba la señal sOkEntryWay1 por parte de pEntryNOut\_Way indicará que la primera entrada principal ha sido exitosamente creada, entonces se procede a crear una salida principal. La interacción de los procesos es muy parecida a la inicialización de la primera entrada principal, solo cambia que el proceso pMainSystemManager solicita al agente pCreatorCardReaderNCamera instanciar un proceso pEntryNOut\_Way y envíe la señal correspondiente a este último para que se le sea asociado un lector de carnés y una cámara. El proceso de inicialización termina cuando el proceso de creación de entradas y salidas principales envía un mensaje al proceso pMainSystemManager llamada sOkCreateOutWay asociando el identificador de dicha salida principal para que sea anexado en su lista de salidas principales.

La figura 6 describe la inicialización de los procesos del bloque ParkingLotSystem, el objetivo de esta interacción es asociar a la zona y al proceso pZoneManager su respectivo controlador. El sistema de parqueo cuenta con controladores de tienen a su mando un conjunto de zonas; dado que el sistema es dinámico es posible que un controlador pueda tener más zonas, ya que el proceso pCtrl no se encuentra en el mismo bloque que pZone se necesita un proceso que sea capaz de crear más zonas, para ello se requiere del proceso pZoneManager. Lo anterior implica que cada proceso pCtrl tendrá asociado hasta *Z* zonas y un único proceso pZoneManager.

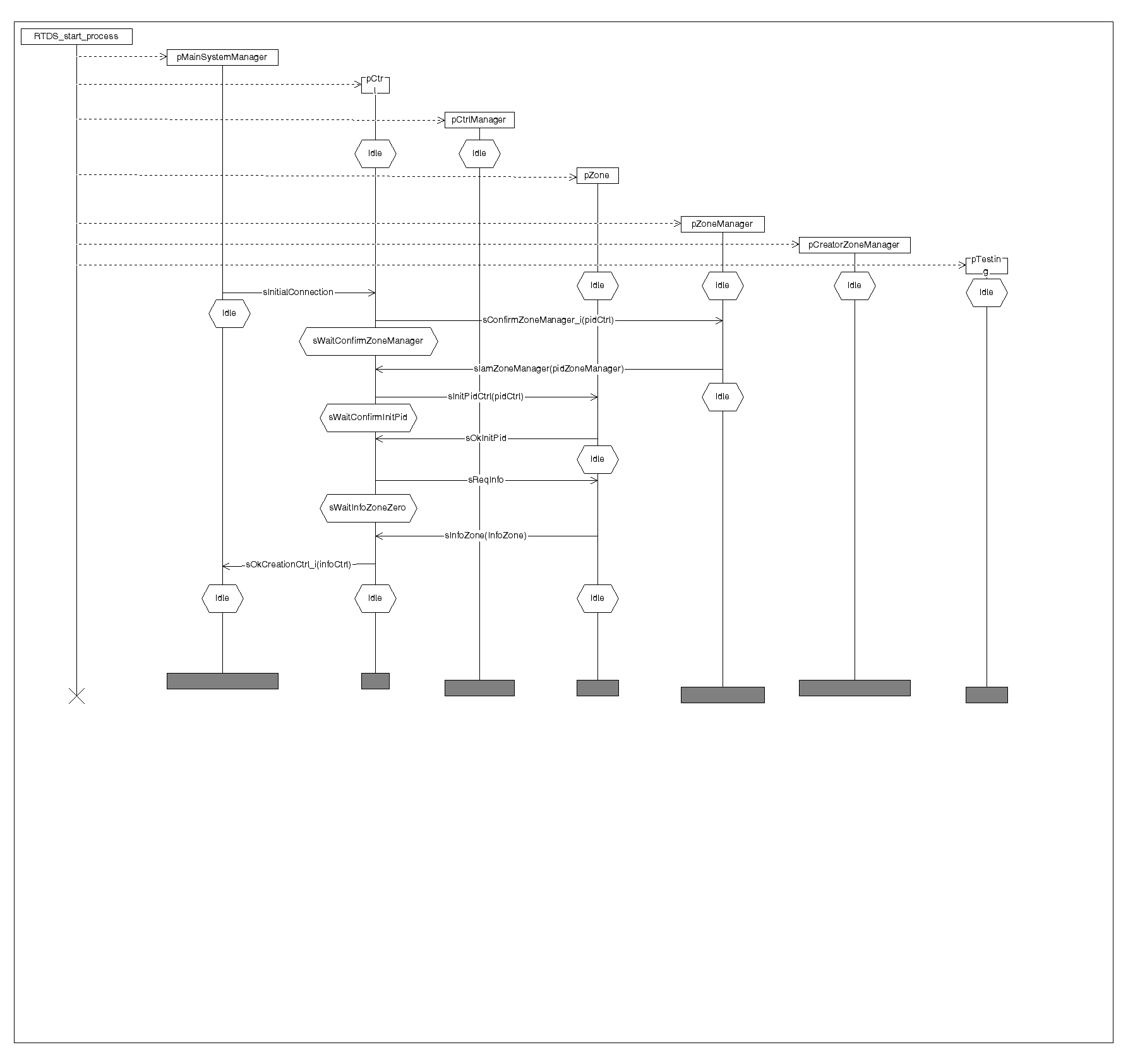


Figura 6. Inicialización de procesos del bloque ParkingLotSystem

La función de cada uno de las señales presentadas a partir de este momento se puede detallar en el Anexo B.

La tabla 4 explica el propósito de cada uno de los MSCs que se diseñaron para la descripción de las especificaciones del sistema de parqueo de la PUJC.

|  |  |
| --- | --- |
| **MSC** | **Descripción** |
| Figura 7. Creación de una entrada principal al sistema de parqueo. | El administrador solicita la creación de una entrada principal al sistema de parqueo. Básicamente se crea el proceso que representa una entrada principal y a éste se le asigna su respectiva cámara y lector de carné. El sistema si ha creado exitosamente la entrada principal, retorna una señal al administrador llamada sOkCreateE\_W. Si el sistema tiene el cupo máximo de entradas principales retornará al administrador la señal sExcEntryWay en lúgar de sOkCreateE\_W y no se efectuará el intercambio de señales para la asignación de cámara y lector de carnés. |
| Figura 8. Creación de una salida principal al sistema de parqueo | El administrador solicita la creación de una entrada principal al sistema de parqueo. Básicamente se crea el proceso que representa una entrada principal y a éste se le asigna su respectiva cámara y lector de carné. El sistema si ha creado exitosamente la entrada principal, retorna una señal al administrador llamada sOkCreateE\_O. Si el sistema tiene el cupo máximo de entradas principales retornará al administrador la señal sExcOutWay en lúgar de sOkCreateE\_O y no se efectuará el intercambio de señales para la asignación de cámara y lector de carnés. |
| Figura 9. Creación de una zona de parqueo cuando el controlador de zonas tiene asociado un pZoneManager | El administrador del sistema envía una señal al proceso pMainSystemManager para crear una zona en un controlador específico, con una inicialización de plazas libres y totales de la zona. Para la solicitud de la zona se usa la señal sAddZone donde su primer parámetro es el número del controlador de zona, el suegundo y tercero son las plazas totales y libres de la zona respectivamente. Si la creación de la zona es exitosa el proceso pMainSystemManager lo hará saber al administrador a través de la señal sOkCreateZone de lo contrario enviará la señal sExcLimitZones, ésta señal se da antes que el proceso pZoneManager haga una instancia del proceso pZone. |
| Figura 10. Creación de una zona de parqueo cuando el controlador de zonas no tiene asociado un pZoneManager | El administrador del sistema envía una señal al proceso pMainSystemManager para crear una zona en un controlador específico. Este escenario es muy parecido al de la figura 9 la diferencia es que el controlador de zonas no tiene un proceso pZoneManager asociado, por lo cual solicita su creación, una vez creado el pZoneManager y asociado a su respectivo controlador de zonas el intercambio de señales respecto a la figura 9 son los mismos. |
| Figura 11. MSC ajustes de parámetros a una zona de parqueo recién creada. | Este MSC representa el intercambio de señales para la inicialización de una zona de parqueo, en este diagrama a la zona se le asigna un controlador y se ajusta su capacidad de aparcamiento de vehículos. |
| Figura 12. MSC creación de un controlador de zonas al sistema de parqueo | El administrador solicita al sistema de parqueo crear un controlador de zonas por medio de la señal sCreateCtrl que recibe el proceso pMainSystemManager. Si la creación es exitosa el proceso pMainSystemManager retornará una señal sOkCreateCtrl al administrador, de lo contrario, hará saber al administrador que no es posible tener más controladores de zonas con la señal sExcLimitCtrl. |
| Figura 13. MSC ingreso de un vehículo al sistema de parqueo. | Este escenario es similar al de la figura 4 presentada anteriormente pero a nivel de procesos. Dado que los procesos pCardReader, pCamera y pDataBase no han sido modelados, se ha colocado un temporizar en la vida del proceso pCamera que se llama timerProcessOCR el cual representa el tiempo de cómputo que tardaría el proceso pCamera para entregar a partir de una foto la placa de un vehículo en un tipo de dato charstring. |
| Figura 14. MSC Salida de un vehículo del sistema de parqueo. | Este escenario es similar al de la figura 13. La dieferencia es que en este escenario el vehóculo está por salir del sistema de parqueo, básicamente el proceso de verificación de usuario habilitado para salir es el mismo que cuando el suaurio está por ingresar con la diferencia que ésta validación se hace a través de la señal sConfirmOutUser. |
| Figura 15. MSC usuario no autorizado intentando ingresar al sistema de parqueo de la | Este escenario representa el caso cuando un usuario desea ingresar al sistema de parqueo de la PUJC pero éste no está habilitado para hacerlo. La dinámica de verificación del usuario es la misma que la figura 13, solo que la respuesta por parte del proceso pDataBase es negativa y no se le da acceso al usuario de ingresar. |
| Figura 16. MSC usuario no autorizado intentando salir del sistema de parqueo de la PUJC. | Este escenario representa el caso cuando un usuario desea salir del sistema de parqueo de la PUJC pero éste no está habilitado para hacerlo. La dinámica de verificación del usuario es la misma que la figura 14, solo que la respuesta por parte del proceso pDataBase es negativa y no se le da acceso al usuario de salir. |
| Figura 17. MSC usuario ingresando a una zona del sistema de parqueo. | Este MSC representa el escenario el cual un vehículo va a ingresar a una zona del sistema. Para saber que un usario ha ingresado se tiene que cumplir la siguiente secuencia: Se interrumpe el primero sensor infrarrojo, sIR1\_Zone, posteriormente el vehóculo interrumpe el sensor infrarrojo 2, sIR2\_Zone y finalmente se verifica que es un vehiculo por la señal capturada en el sensor inductivo sLoopInductive\_Zone. La zona reportará que ha ingresado un vehículo a su respectivo controlador con la señal sEntered\_Car. Otra posibilidad que es válida para el ingreso del vehiculo es cuando se interrumpe el sensor infrarrojo 4, luego el sensor infrarrojo 3 y finalmente la recepción de la señal del loop inductivo. |
| Figura 18. MSC usuario saliendo de una zona del sistema de parqueo. | Este MSC representa el escenario el cual un vehículo va a salir de una zona del sistema. Para saber que un usario ha ingresado se tiene que cumplir la siguiente secuencia: Se interrumpe el primero sensor infrarrojo 3, sIR3\_Zone, posteriormente el vehíulo interrumpe el sensor infrarrojo 4, sIR4\_Zone y finalmente se verifica que es un vehiculo por la señal capturada en el sensor inductivo sLoopInductive\_Zone. La zona reportará que ha salido un vehículo de su zona a su respectivo controlador con la señal sOut\_Car. Otra posibilidad que es válida para reconocer que un vehiculo va a salir de la zona es cuando se interrumpe el sensor infrarrojo 2, luego el sensor infrarrojo 1 y finalmente la recepción de la señal del loop inductivo. |
| Figura 19. MSC Ingreso y salida de un vehículo a través del proceso Testing | Este MSC representa la entrada y salida de vehículos en una zona del sistema, imagen lado izquierod y derecho respectivamente. La diferencia de este escenario a los de la figura 17 y 18 es que las señales de los sensores son simuladas desde el proceso pMainSystemManager. El controlador y la zona que se quiere entrar y sacar vehículos se da a través del proceso temporal pTesting. Cabe destacar que las señales de sensores generadas por el proceso pMainSystemManager no deben ser considerados en la implementación del sistema de parqueo. |

Tabla 4. Descripción de los MSC correspondientes a la especificación del sistema.

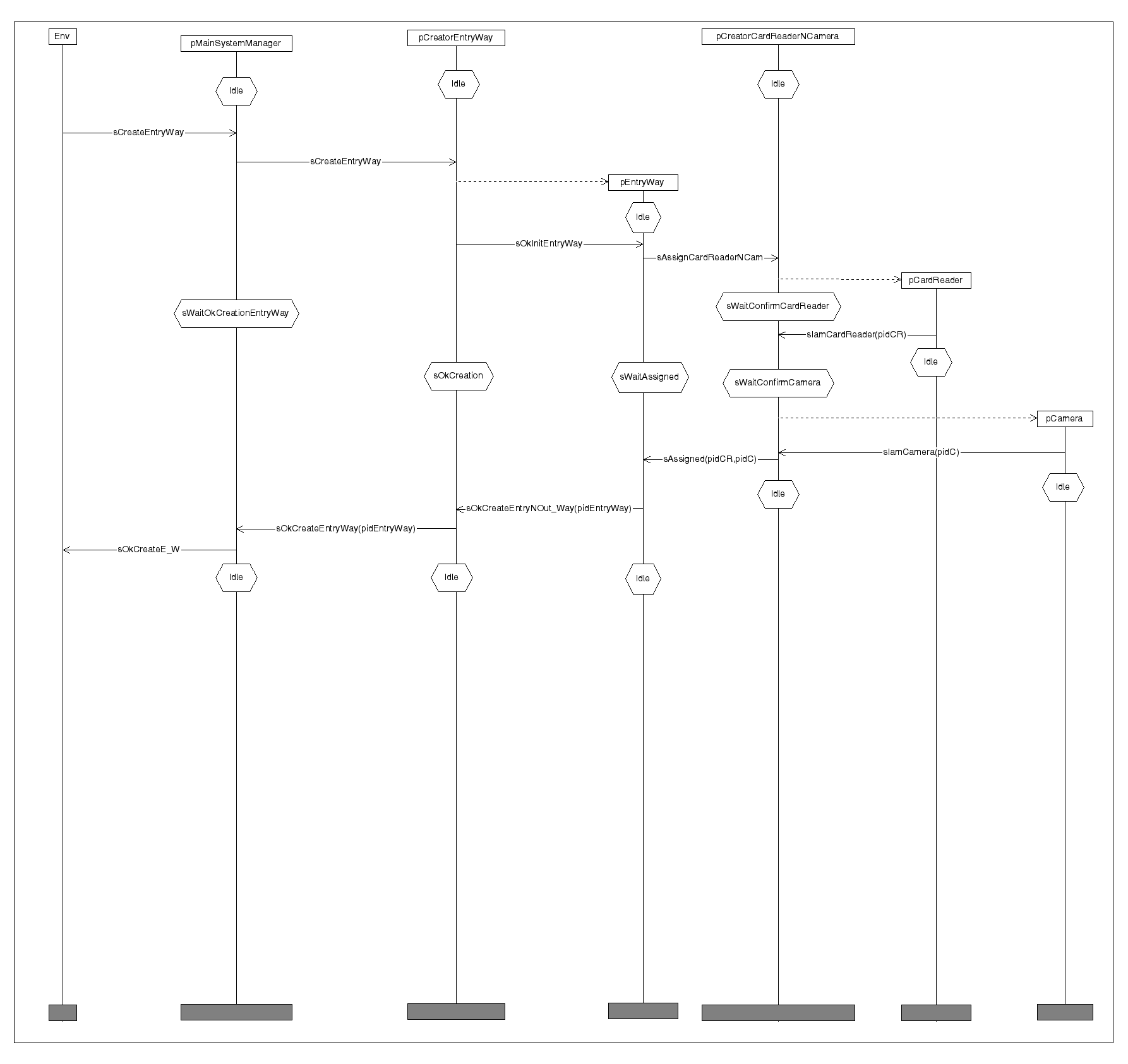


Figura 7. Creación de una entrada principal al sistema de parqueo.

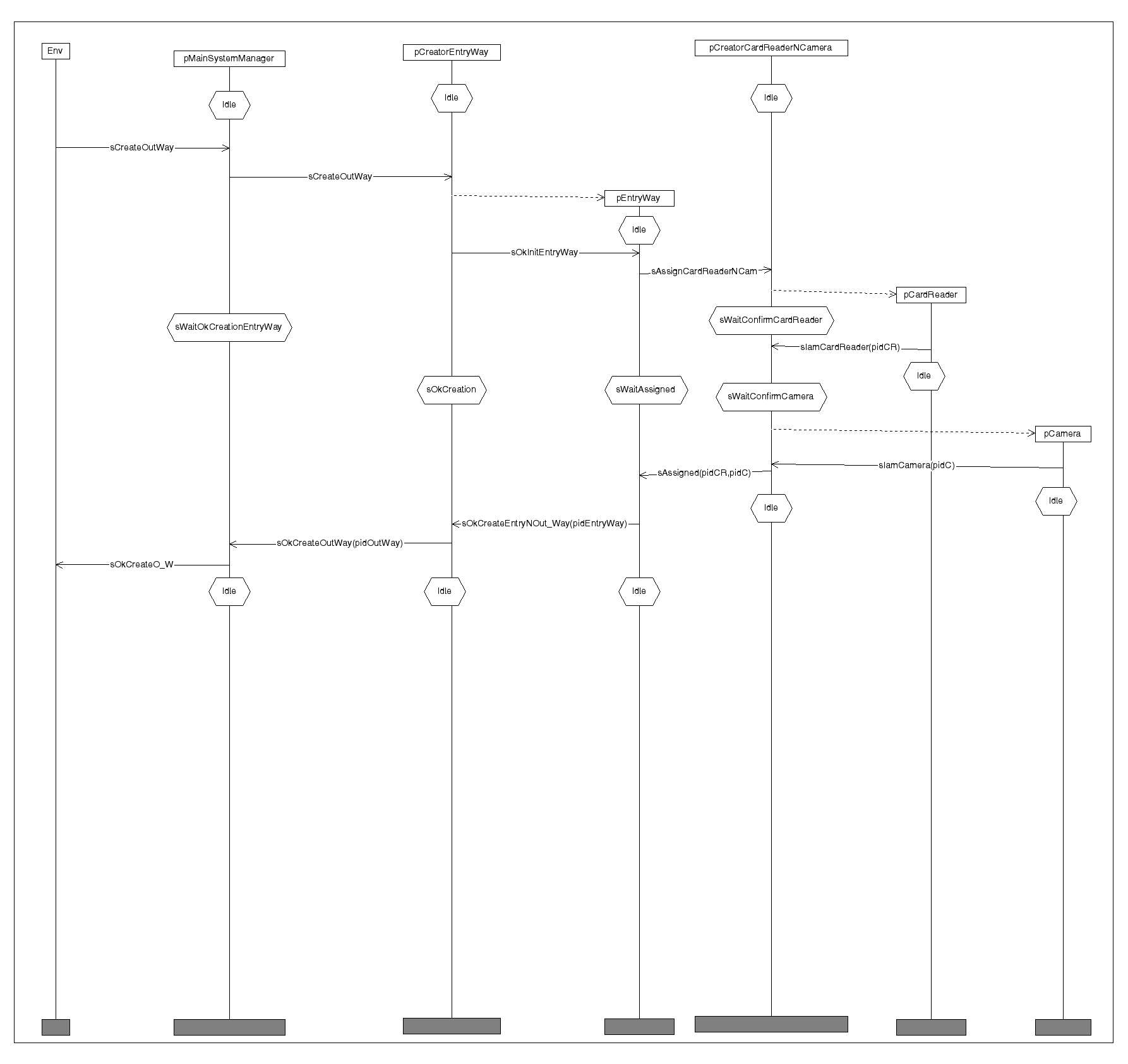


Figura 8. Creación de una salida principal al sistema de parqueo

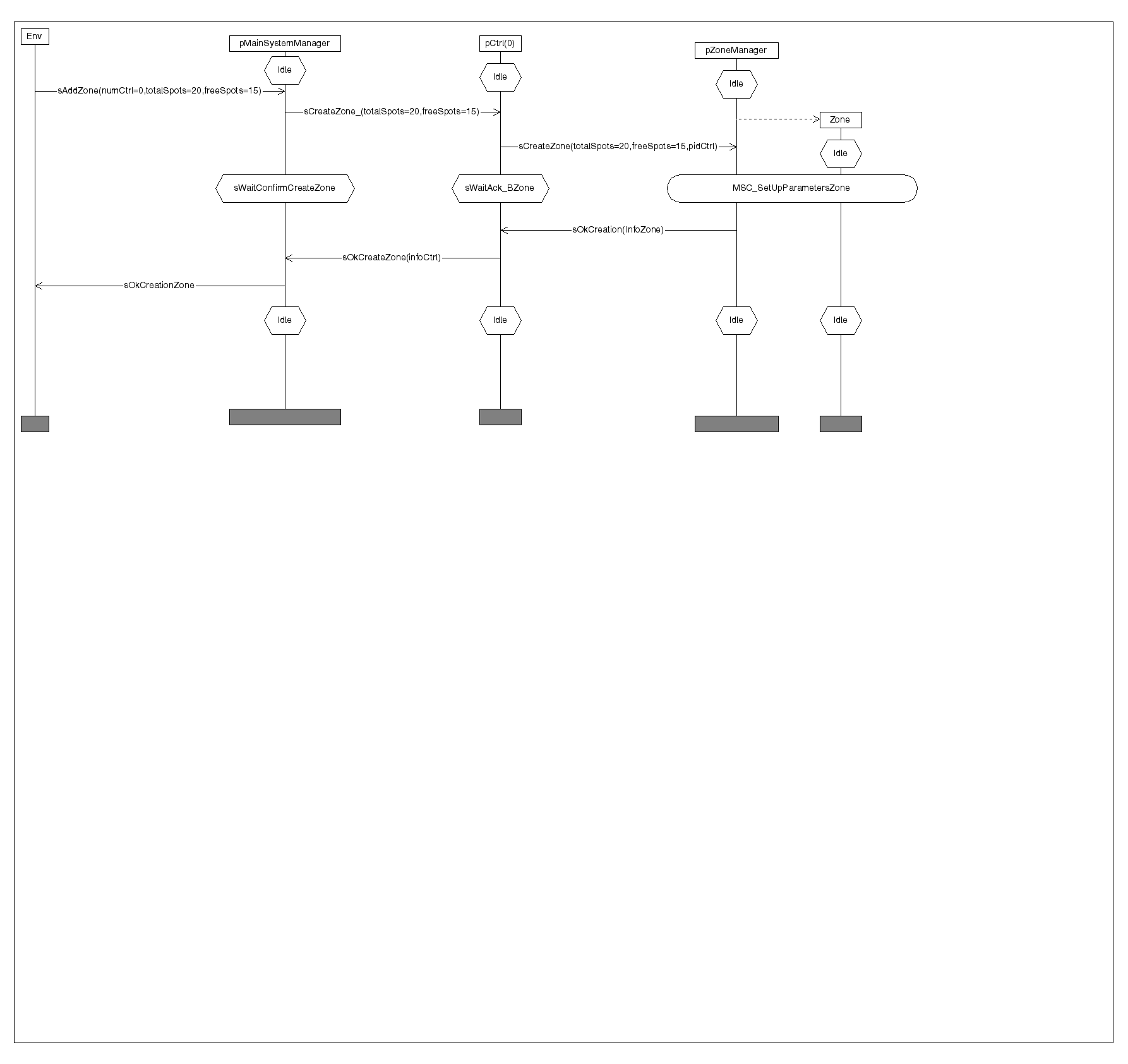


Figura 9. Creación de una zona de parqueo cuando el controlador de zonas tiene asociado un pZoneManager

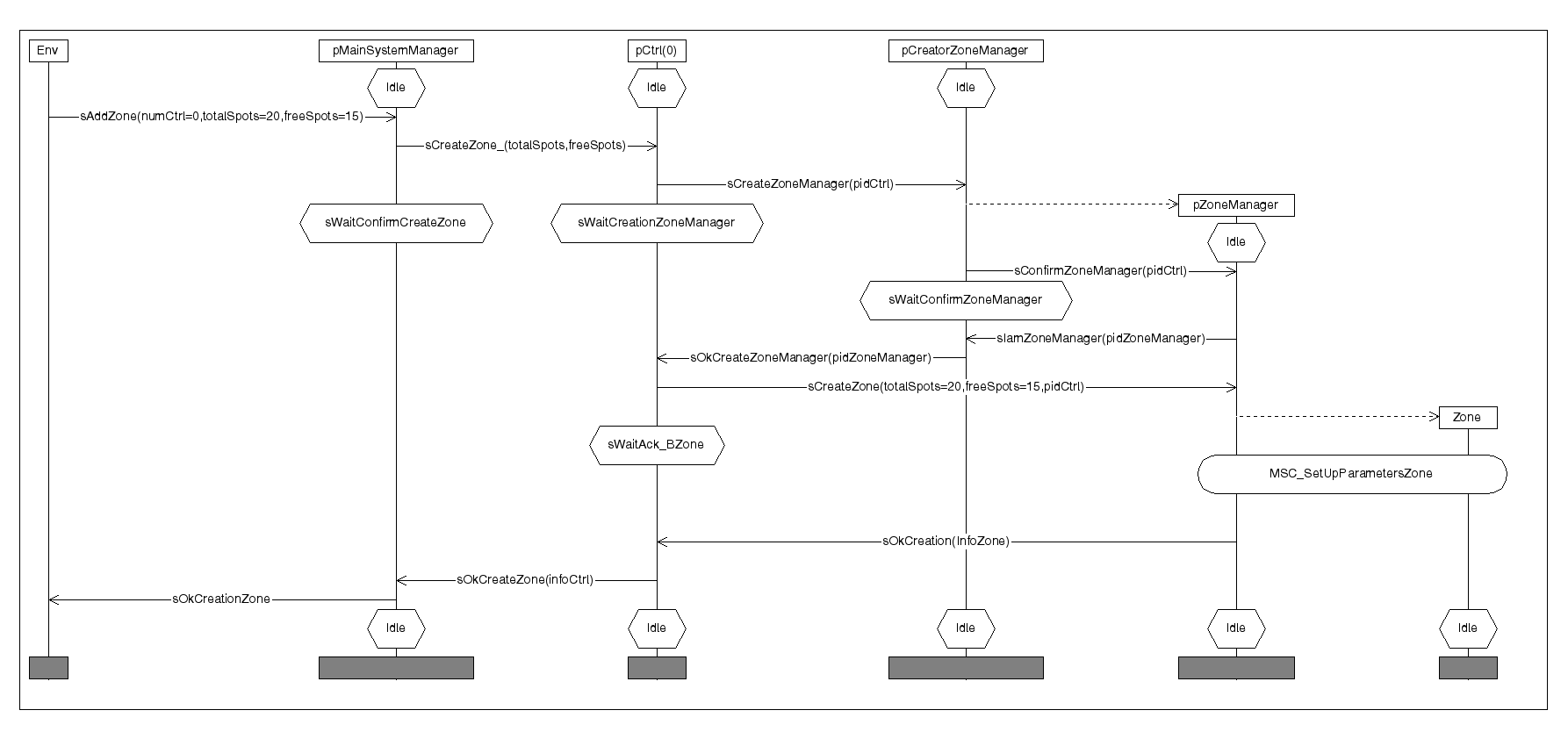


Figura 10. Creación de una zona de parqueo cuando el controlador de zonas no tiene asociado un pZoneManager

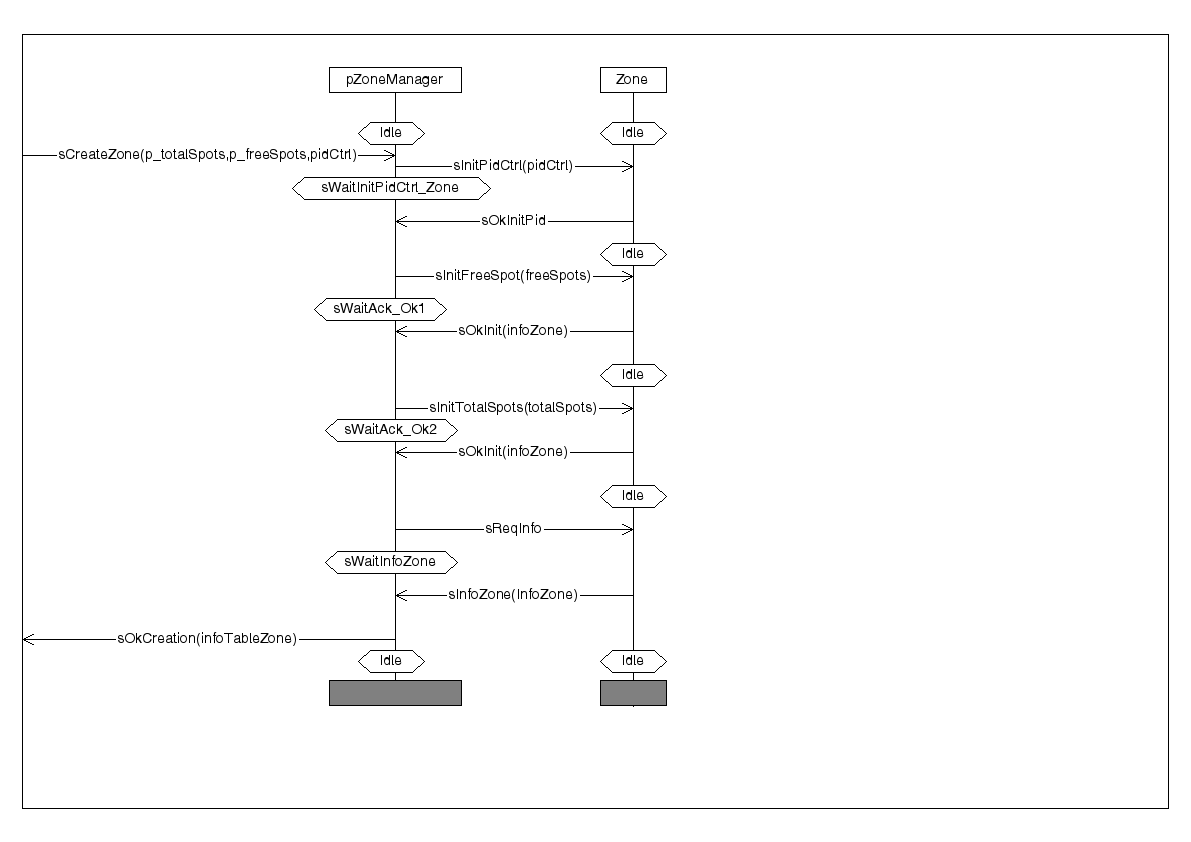


Figura 11. MSC ajustes de parámetros a una zona de parqueo recién creada.

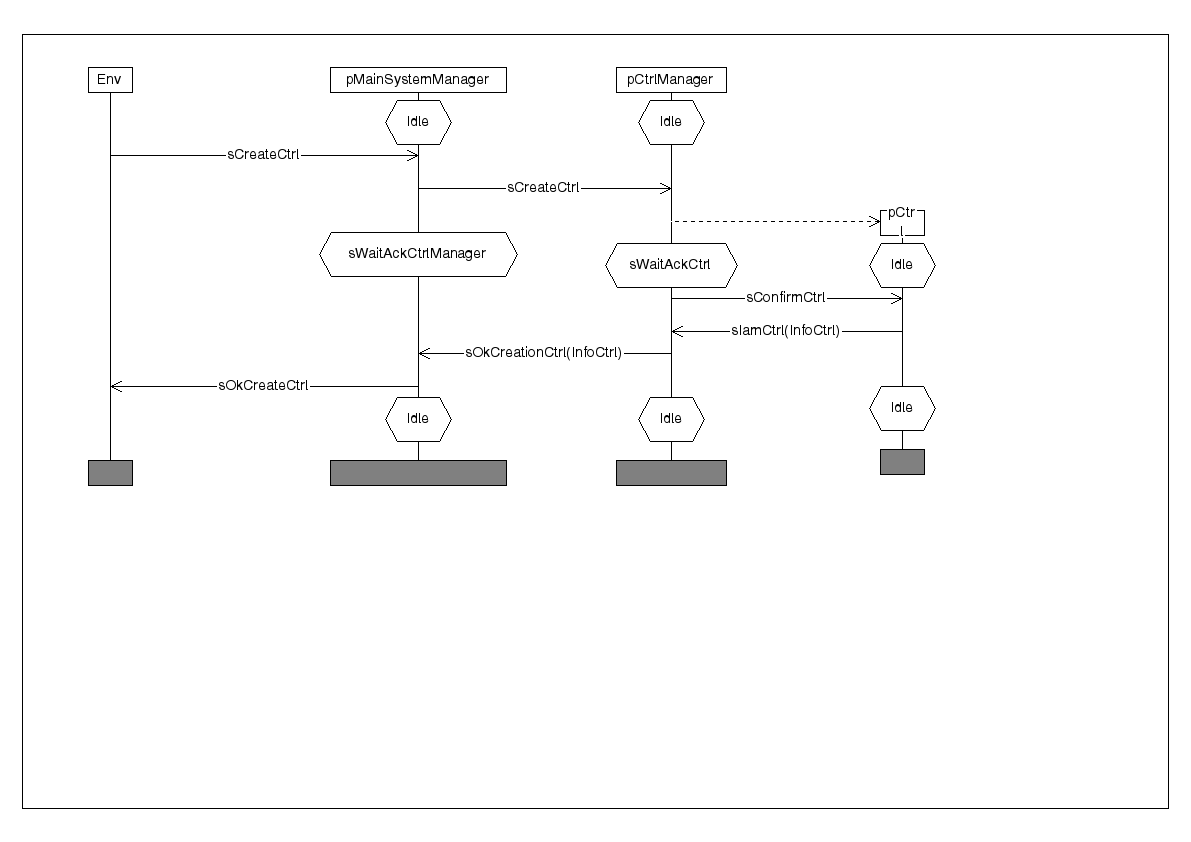


Figura 12. MSC creación de un controlador de zonas al sistema de parqueo.

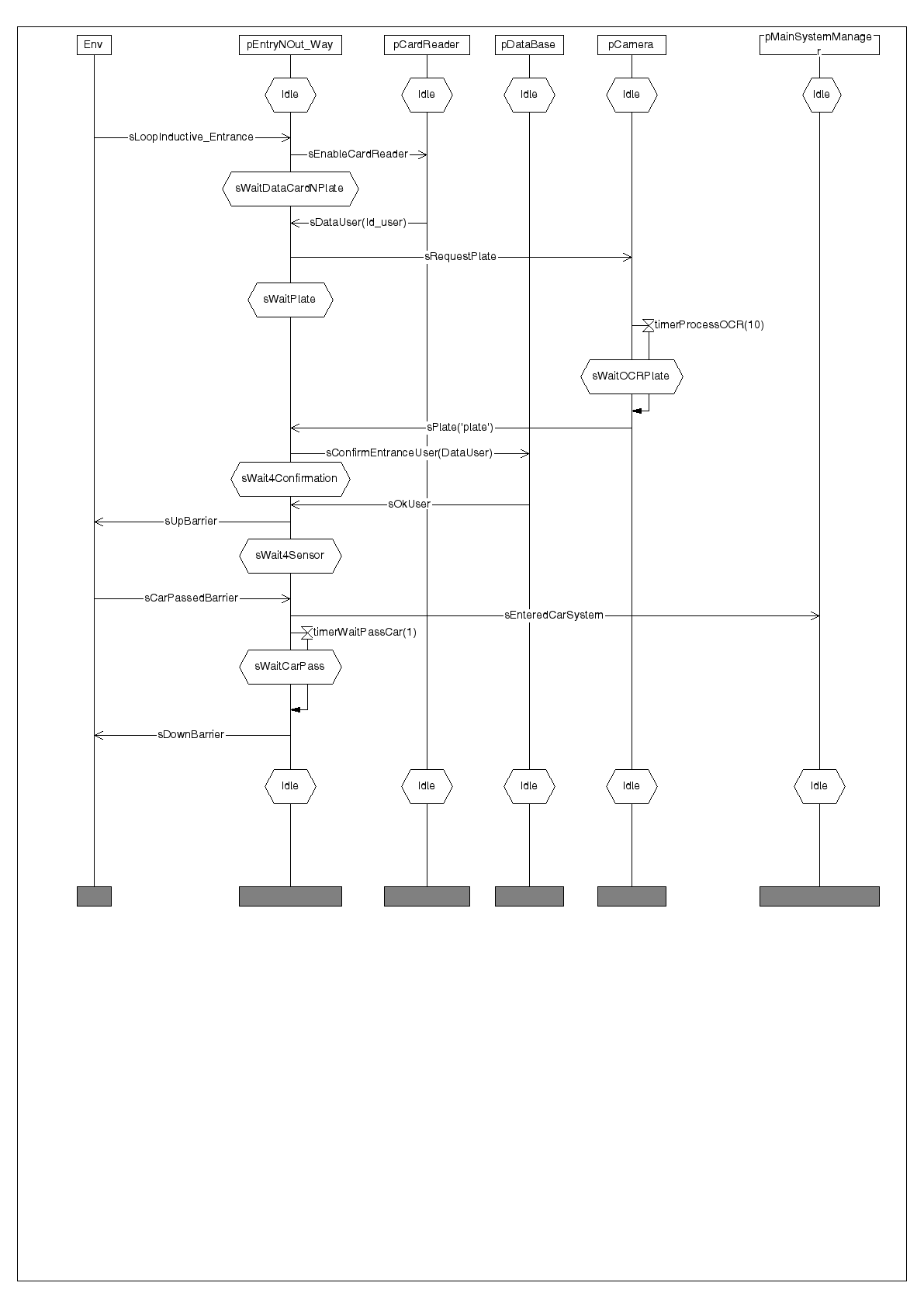


Figura 13. MSC ingreso de un vehículo al sistema de parqueo.

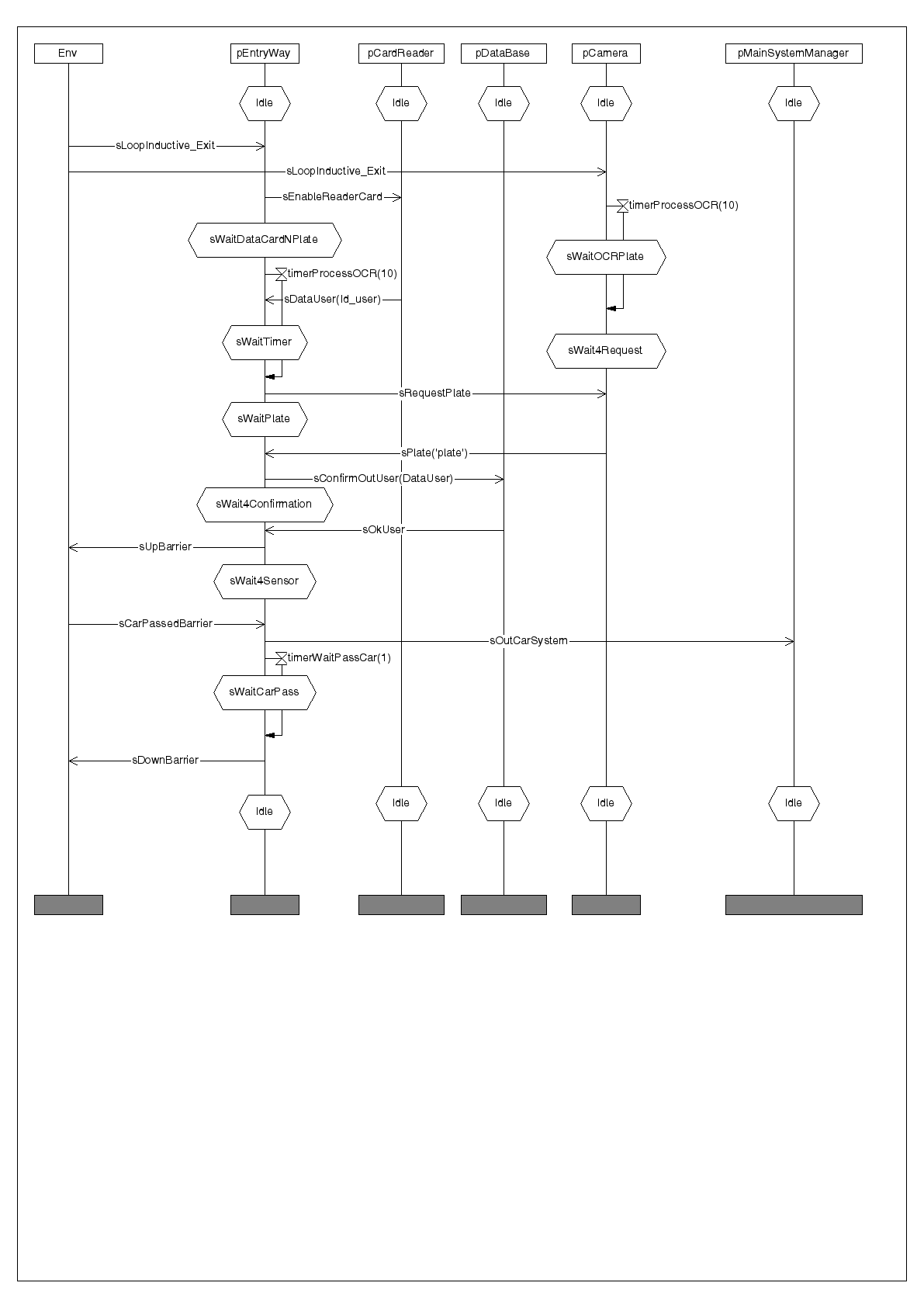


Figura 14. MSC Salida de un vehículo del sistema de parqueo.

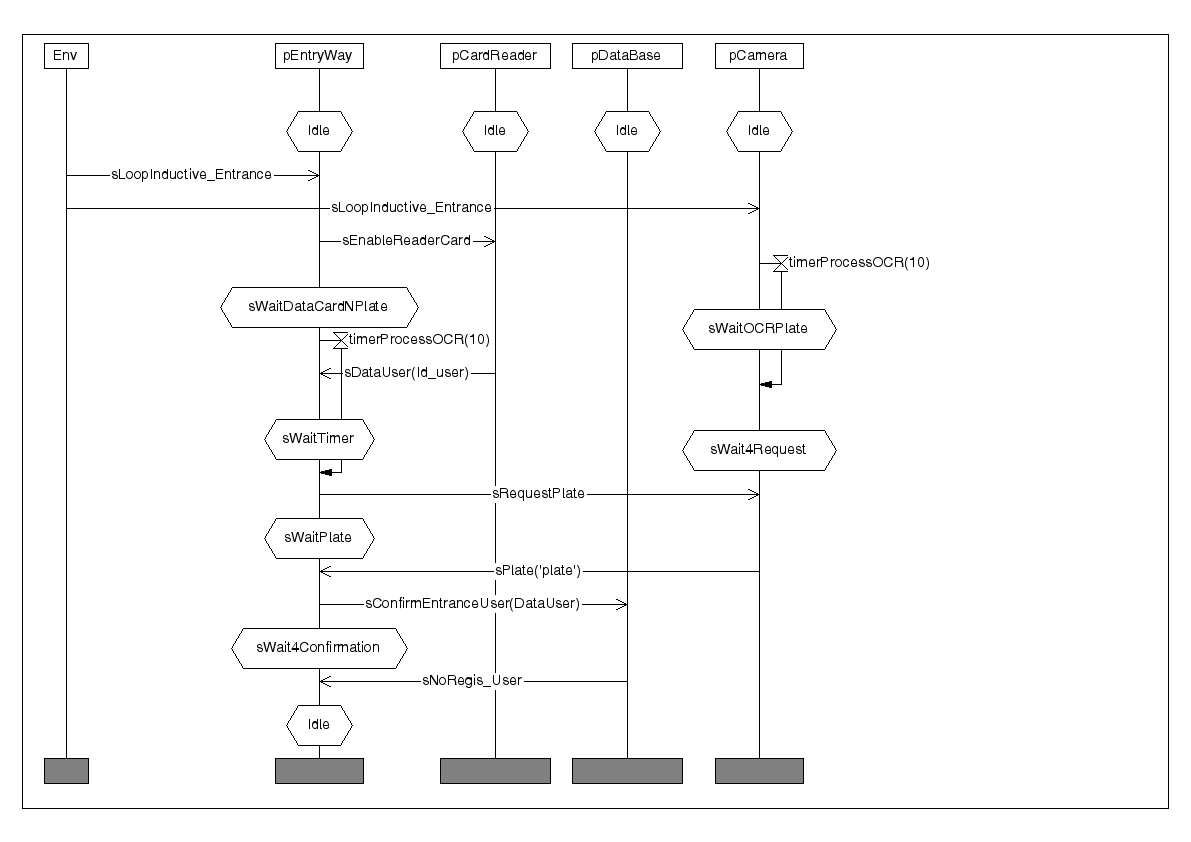


Figura 15. MSC usuario no autorizado intentando ingresar al sistema de parqueo de la PUJC.

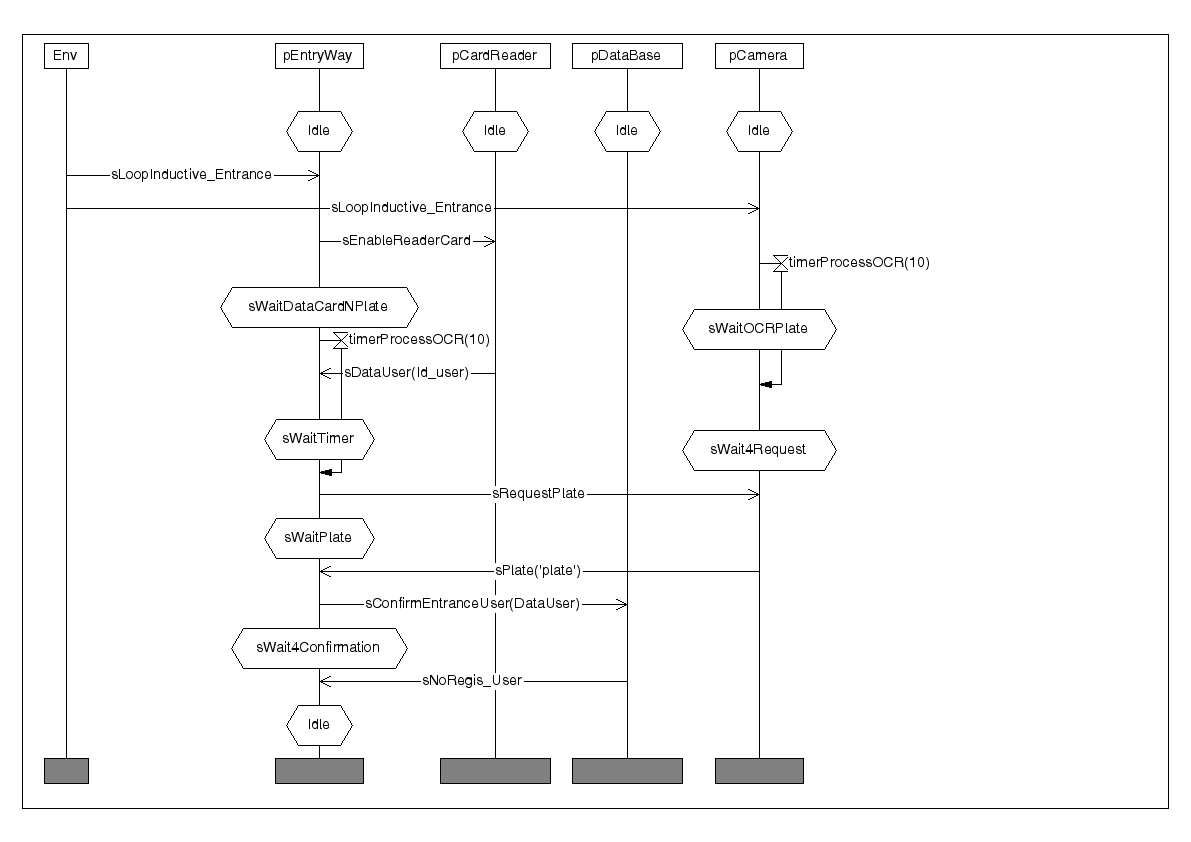


Figura 16. MSC usuario no autorizado intentando salir del sistema de parqueo de la PUJC.

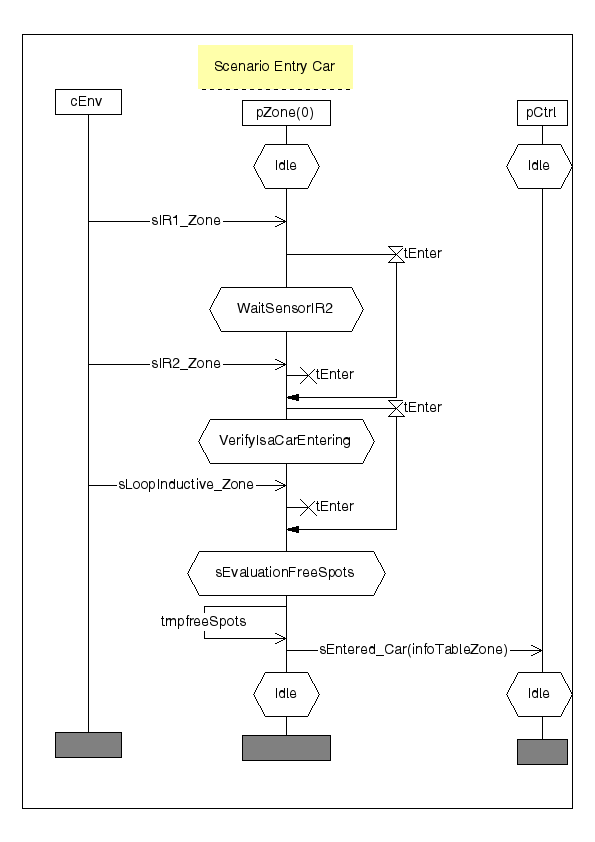


Figura 17. MSC usuario ingresando a una zona del sistema de parqueo.

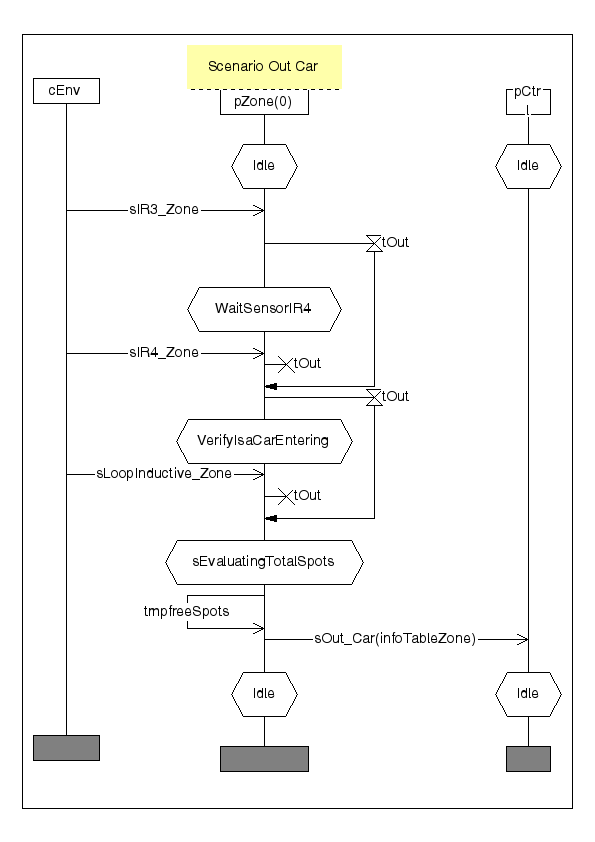


Figura 18. MSC usuario saliendo de una zona del sistema de parqueo.

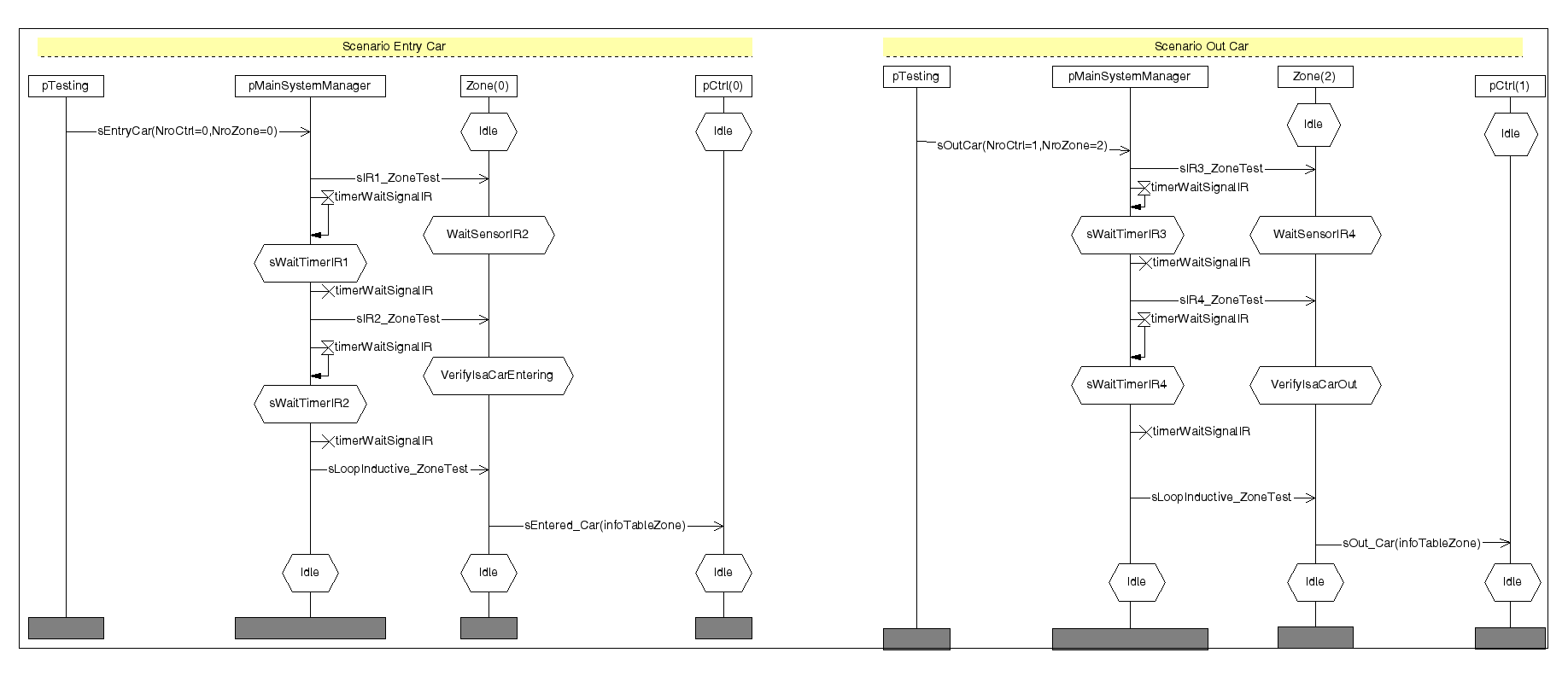


Figura 19. MSC Ingreso y salida de un vehículo a través del proceso Testing´

## Implementación de las especificaciones en SDL

La estrategia de diseño usada para el desarrollo del sistema es Bottom-Up. Se inicia implementando el bloque BZone representado en la figura 20, dado que contiene procesos que poseen un alto nivel de abstracción. Inicialmente se diseñó el proceso pZone el cual representa una zona de parqueo.

Como se había planteado en la especificación del sistema ver la tabla 1, el administrador puede crear una zona e inicializar sus parámetros los cuales son: plazas totales y libres. Ambos parámetros son subtipos del tipo de dato integer, ver Anexo A para los tipos de datos usados en el modelado del sistema de parqueo. La zona tiene la facultad de reportar a su respectivo controlador el ingreso o salida de vehículos de ésta, y de enviar su información como una estructura de datos InfoZone, ver Anexo A, cada que haya un requerimiento por parte del controlador.

La figura 21, 22 y 23 representan la máquina de estados que se diseñó para detectar si un vehículo está por ingresar o salir de una zona del sistema de parqueo. Como se aprecian la figura 22, cuando la zona no tenga plazas libres, freeSpots, el proceso pZone reportará a su respectivo proceso pCtrl que tiene cero plazas libres, así ingresen más vehículos a esta zona de esta forma evitará enviar valores negativos de plazas libres. Lo anterior es similar a la figura 23 cuando la zona tenga todas sus plazas libres e intente salir un vehículo de ésta zona, se reportará que las plazas libres son iguales a las máximas permitidas o ajustadas por el administrador evitando enviar valores mayores a los permitidos por él sistema.

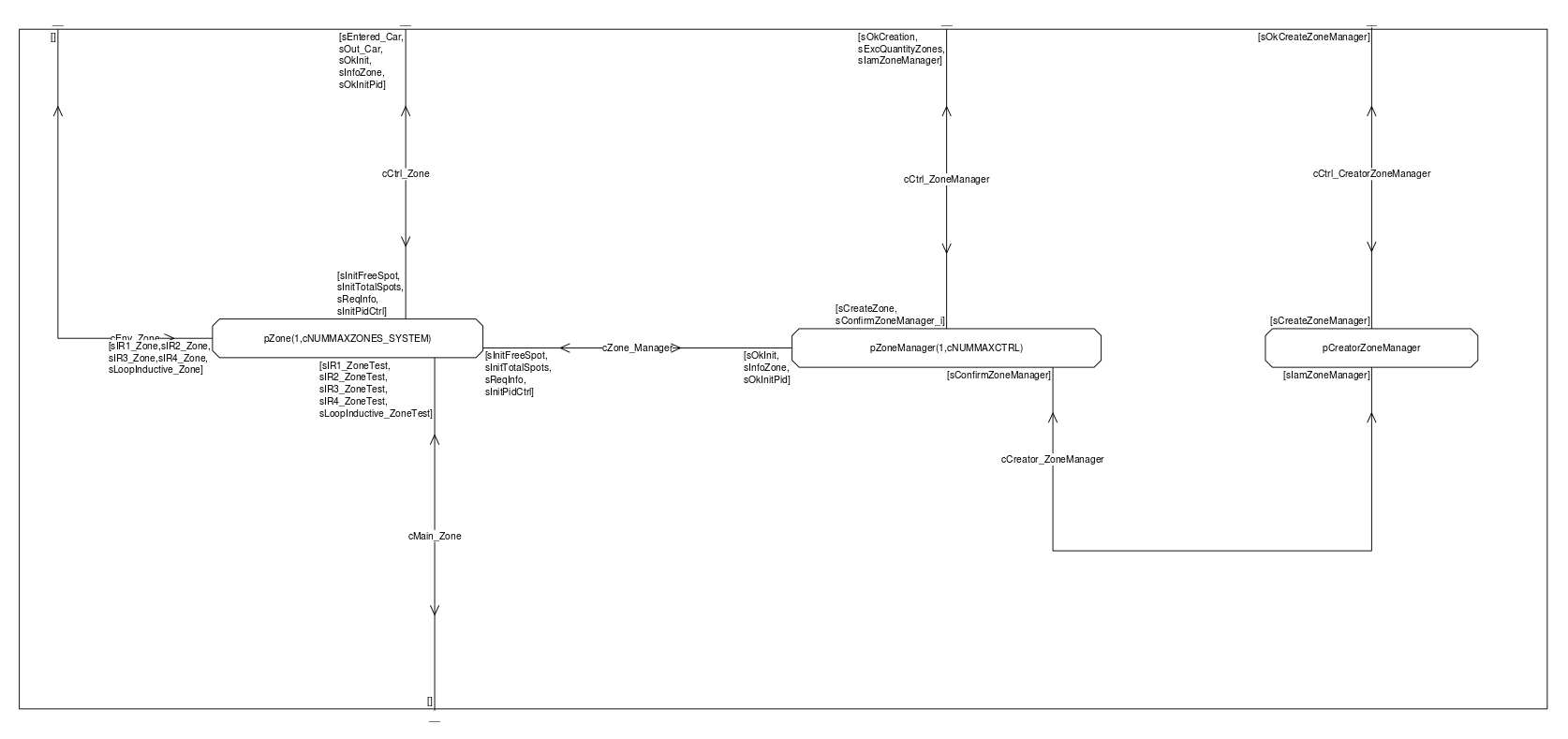


Figura 20. Arquitectura del bloque BZone

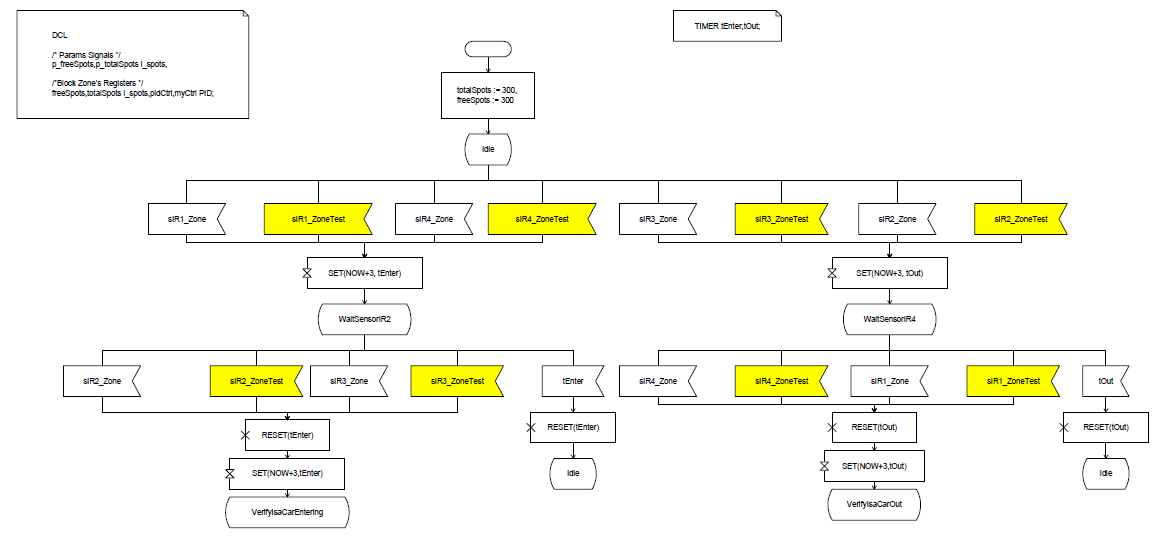


Figura 21. Máquina de estados del proceso pZone para el acceso o salida de un vehículo en una zona.

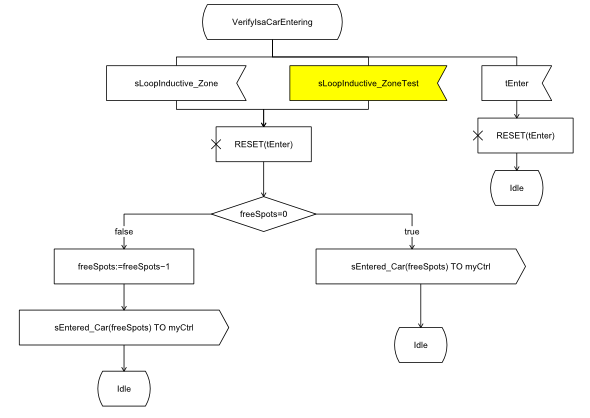


Figura 22. Continuación máquina de estados para el ingreso de un vehículo en una zona de parqueo.

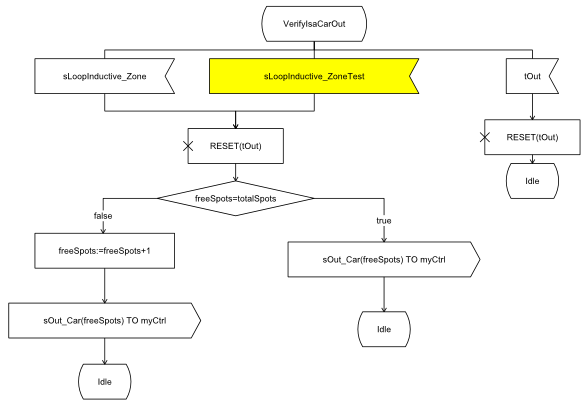


Figura 22. Continuación máquina de estados para la salida de un vehículo de una zona de parqueo.

Las señales de entrada resaltadas que se encuentran en la figura 21, son provenientes del proceso pMainSystemManager, nuevamente éste tipo de señales son usadas para implementar pruebas funcionales de caja negra, su utilidad se explicará en la siguiente sección.

La figura 21 muestra la implementación de dos temporizadores que tienen como función regresar al proceso pZone a un estado válido y que éste no se quede en uno de bloqueo. Un ejemplo en el cual éstos temporizadores serían útiles es cuando al proceso pZone reciba la señal del sensor sIR1\_Zone y posteriormente la señal sIR2\_Zone, lo anterior se consideraría que un vehículo está por ingresar a dicha zona, pero si el causante de la interrupción fue una persona nunca llegaría la señal sLoopInductive\_Zone, por lo cual el proceso pZone quedaría en el estado VerifyIsaCarEntering, si ésta señal nunca llega después de cierto tiempo el temporizador coloca al proceso nuevamente en el estado Idle dónde puede efectuar otras funciones sin bloqueo alguno.

La figura 24 se muestran los estados y las transiciones que el proceso pZone efectúa para la inicialización de plazas totales, plazas libres, requerimientos de información e inicialización del identificador de su controlador.

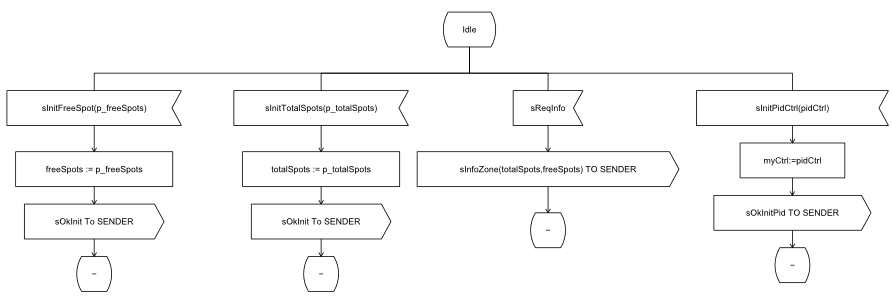


Figura 24. Máquina de estados para la inicialización de parámetros de una zona y solicitud de requerimiento de información a ésta.

Una vez implementada la máquina de estados del proceso pZone, se implementa el proceso pZoneManager que está encargado de instanciar más procesos pZone y asignarle a éste su respectivo controlador de zonas.

En las figuras 25 y 26 se pueden apreciar la máquina finita del proceso pZoneManager, inicialmente el sistema permanece en el estado Idle y está habilitado para recibir las siguientes señales: sCreateZone, sConfirmZoneManager y sConfirmZoneManager\_i. La primera señal tiene asociado dos parámetros de tipo i\_spots que son subtipos del tipo de dato Integer, básicamente cuando el proceso pZoneManager recibe esta señal crea un instancia de una zona, si no hay capacidad enviará una señal sExcQuantityZones a su respectivo controlador indicando que no es posible anexar más zonas, de lo contrario creará un proceso pZone y efectuará el intercambio de señales con éste para inicializar sus parámetros.

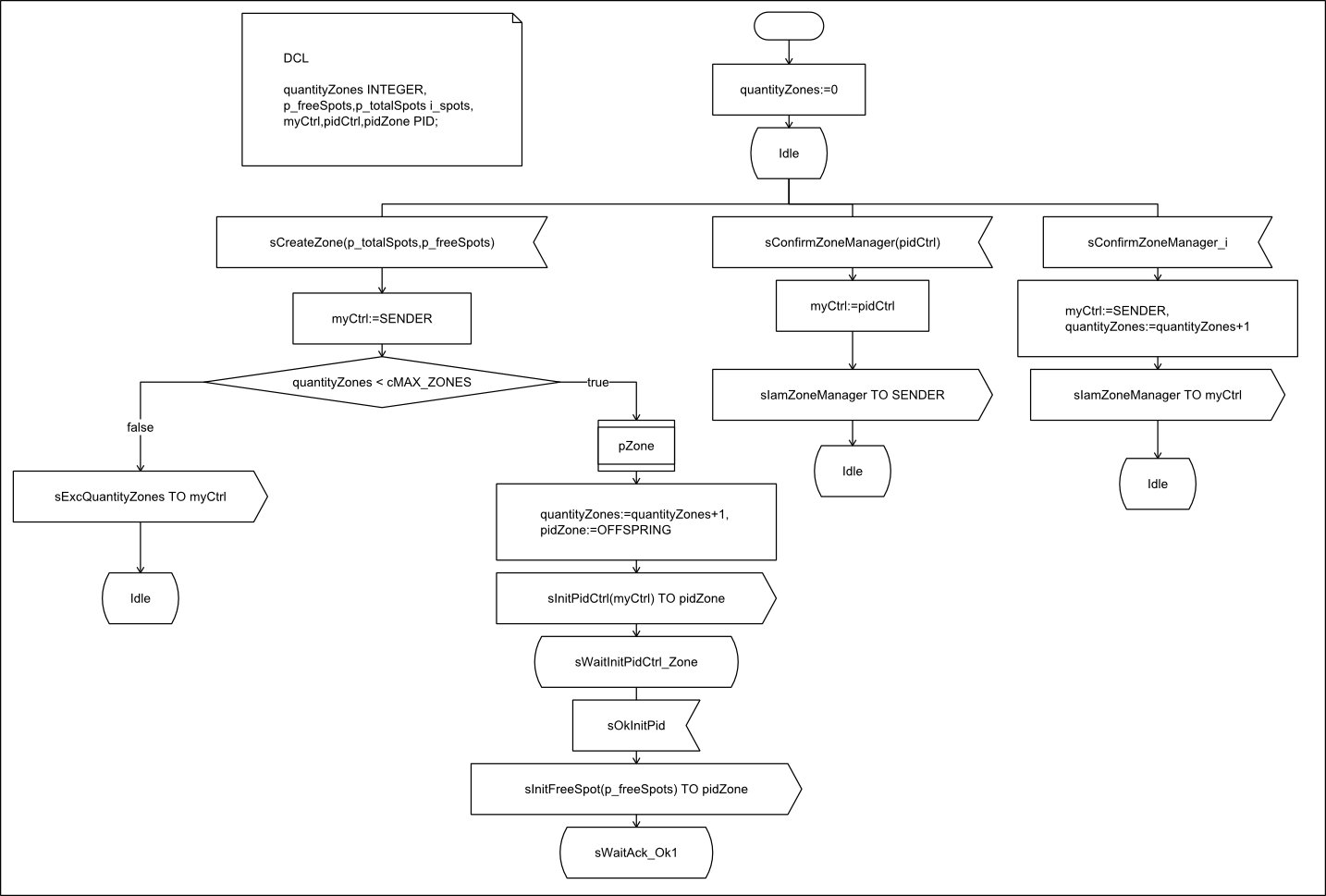


Figura 25. Máquina de estados del proceso pZoneManager para la creación de zonas e inicialización de su respectivo controlador de zonas.

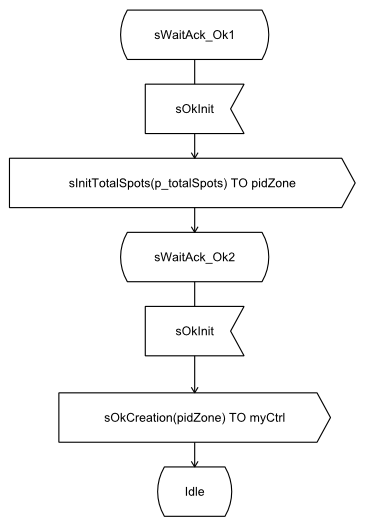


Figura 26. Continuación máquina de estados para la creación de una zona de parqueo.

La figura 27 representa la máquina de estados finita del proceso pCreatorZoneManager, la única función que tiene este proceso es de instanciar procesos pZoneManager y asignarle a este su respectivo controlador de zonas.

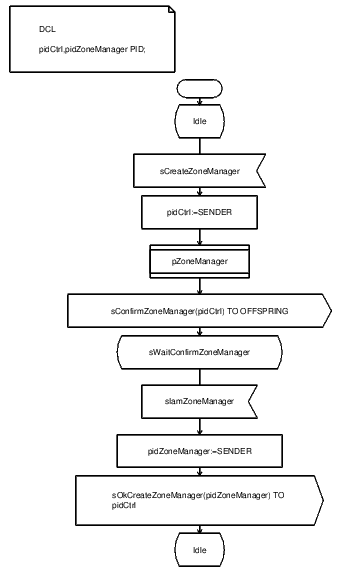


Figura 27. Máquina de estados finita del proceso pCreatorZoneManager.

La figura 28 muestra la arquitectura del bloque UnmodeledProcesses. En esta arquitectura se puede apreciar que los procesos pCardReader que representa el lector de carnés y pCamera que representa la cámara, solicitan al ambiente o entorno los datos que deberían de entregar al proceso pEntryNOut\_Way, de esa forma éstos procesos solo sirven para mostrar los canales y las conexiones a otros procesos que deberían de ser usados en el momento de modelarlos.

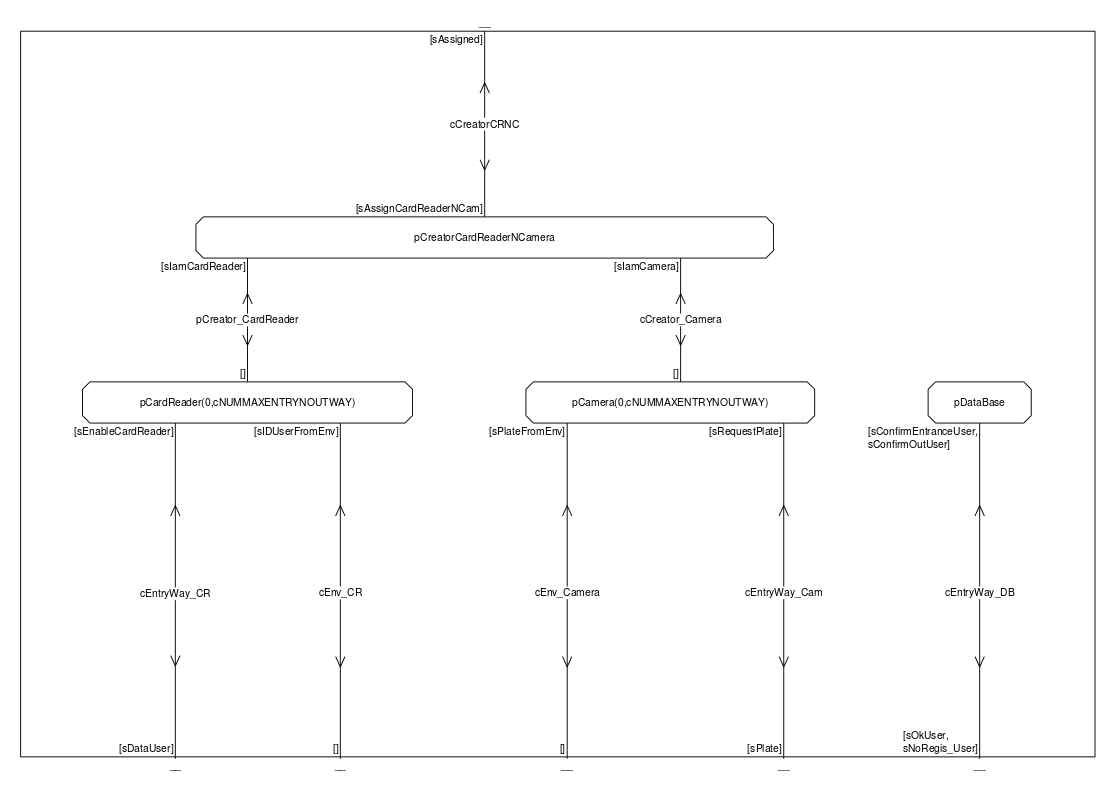


Figura 28. Arquitectura del bloque Unmodeled Processes

Para el modelado del repositorio de información se ha construido un arreglo de datos que contiene un identificador que tiene asociada una única placa, actualmente el sistema cuenta con 40 usuarios registrados, pero no se limita a esta cantidad en su implementación ni tampoco se desvía del objetivo de este trabajo de grado. La figura 29 representa la máquina de estados del proceso pDataBase que representa los repositorios de información del sistema.

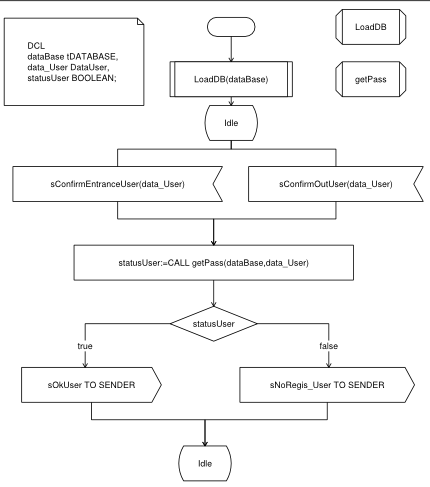


Figura 29. Máquina de estados del proceso pDataBase

Las figuras 30 representan los procedimiento utilizados en la máquina de estados del proceso pDataBase, y la figura 31 representa el procedimiento que valida si un usuario tiene acceso al sistema de parqueo, si su identificador obtenido del lector de carnés y la placa tomada por la cámara se encuentran en el repositorio. Este tipo de búsquedas se podrían ampliar estableciendo algunas políticas de acceso por ejemplo:

* Que un usuario con su identificador pueda tener asociado varias placas de vehículos.
* Que el usuario tenga acceso al sistema si y solo si éste vehículo aparece en el repositorio que ya ha salido.
* Que un mismo usuario no pueda tener acceso al parqueo con otro vehículo registrado si ya a un ingresado otro.

Las opciones descritas anteriormente serían parte de las especificaciones para el modelado y diseño de los repositorios del sistema de parqueo.

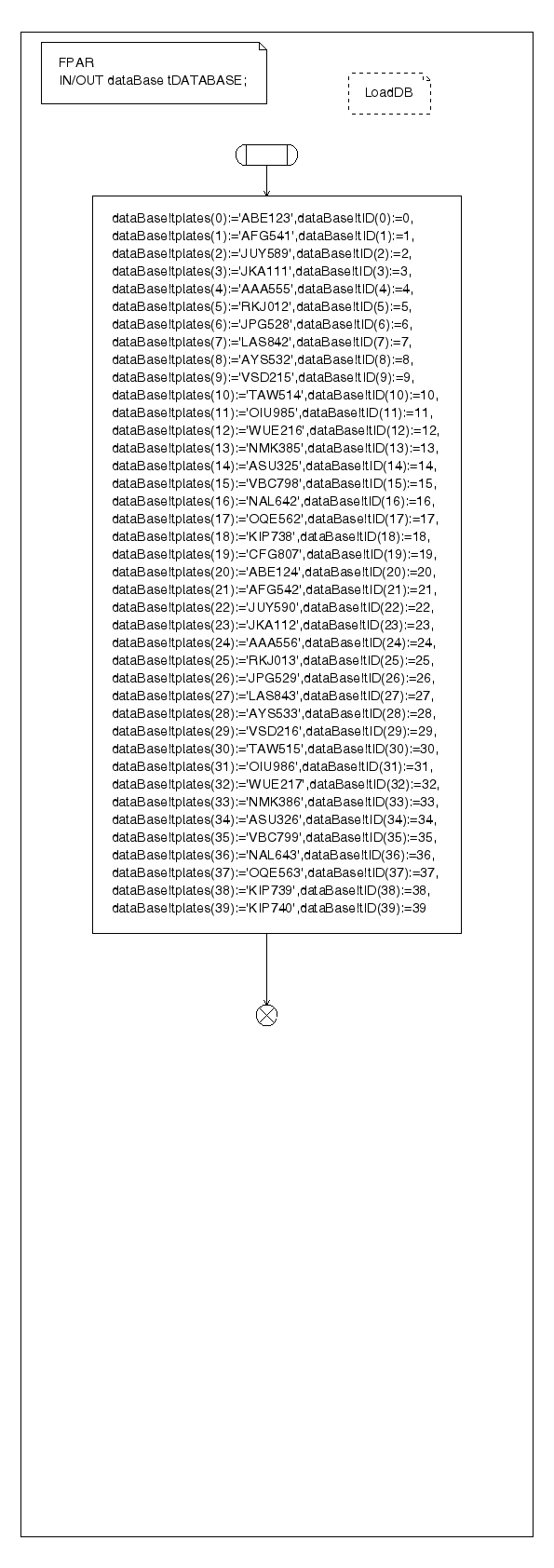


Figura 30. Procedimiento para cargar los datos al repositorio del sistema de parqueo.

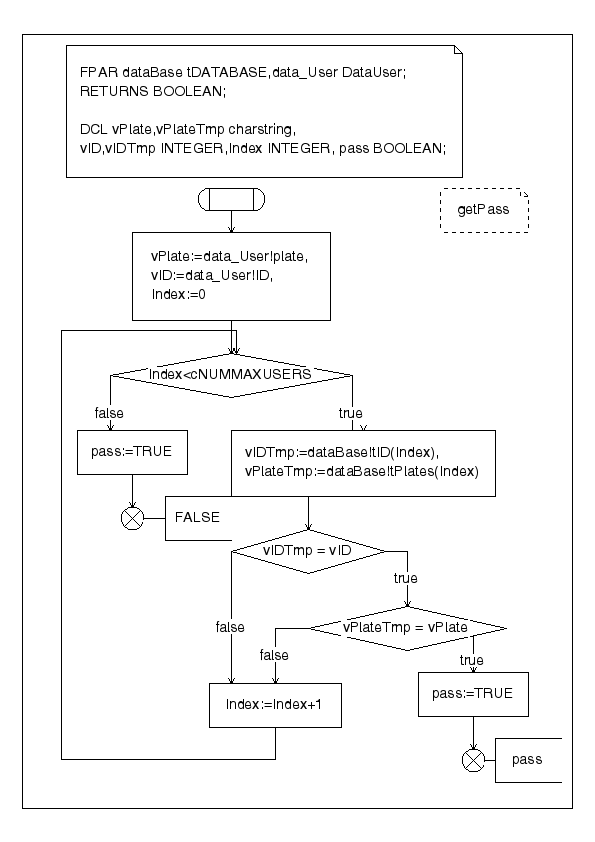


Figura 31. Procedimiento para validar si un usuario tiene acceso o salida del sistema de parqueo.

Para más información del diseño de las otras máquinas de estado véase el Anexo C.

## Pruebas

Al igual que las estrategias de diseño mencionados en la sección del modelado, éstas pueden ser empleadas en la fase de pruebas de un modelo, en este caso se va a hacer uso de la estrategia Bottom-Up. La herramienta RTDS posee un módulo que permite hacer pruebas funcionales de caja negra, la forma que se puede evaluar si una prueba es satisfactoria en dicha herramienta es comparando el MSC que genera cada estímulo al sistema. Las pruebas en RTDS se pueden implementar en una interfaz gráfica como se explicará más adelante.

Una de las ventajas que tiene el módulo de pruebas de RTDS es que permite evaluar un modelo a través de un escenario específico de forma rápida y sencilla, pero carece de automatización en cuanto a los estímulos inyectados al sistema bajo prueba. Por otro lado el lenguaje TTCN-3 ha sido diseñado para hacer pruebas funcionales de caja negra y tiene la ventaja de poder automatizarlas generando una mayor cantidad de estímulos, lo que se traduce a obtener sistemas más seguros.

La figura 32 muestra la interfaz gráfica de la herramienta RTDS para probar modelos descritos en SDL.

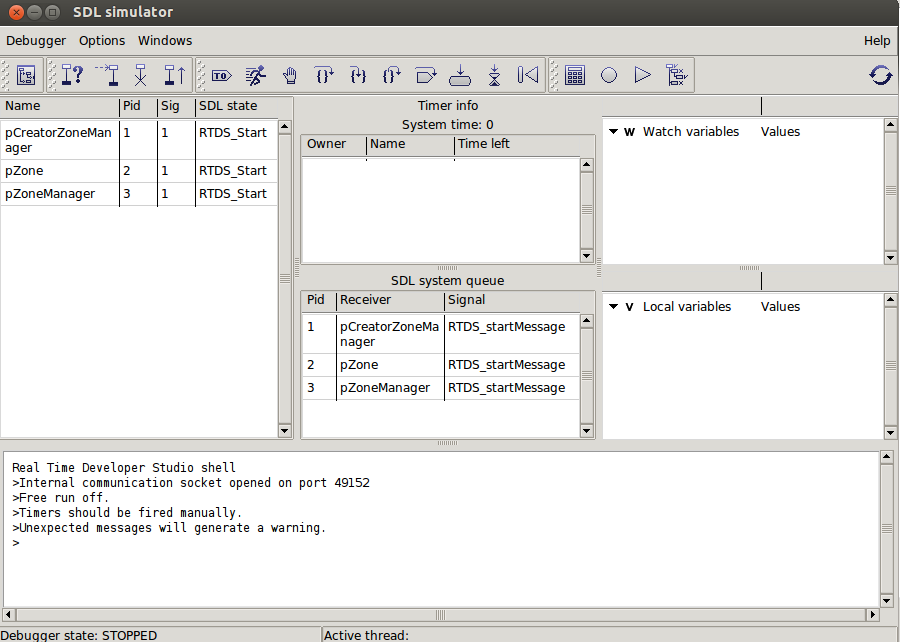


Figura 32. Interfaz gráfica para el módulo de pruebas en la herramienta RTDS

En el botón **TO** se puede seleccionar las señales que se desean enviar a procesos específicos, como se muestra en la figura 33.

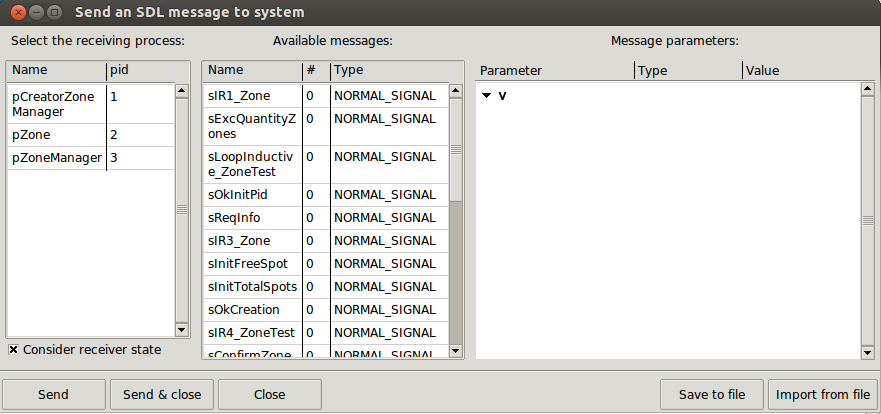


Figura 33. Interfaz envío de señales a procesos a través de la herramienta RTDS.

Como se describió en la parte del modelado del sistema de parqueo, el bloque BZone tiene tres procesos entre los cuales se encuentra el proceso pCreatorZoneManager que está encargado de instanciar agentes de tipo pZoneManager. La figura 34 muestra el resultado de dicha prueba, la forma de verificar que el diseño de las máquinas de estados quedó conforme a los requerimientos es por medio de un MSC que es generado por medio de la interfaz gráfica de pruebas de la herramienta RTDS, para más información de ésta herramienta véase el manual de referencia[[4]](#footnote-4).

Lo que nos dice la figura 34 es básicamente que el proceso pCreatorZoneManager incialmente se encuentra en el estado Idle y está esperando por la señal sCreateZoneManager, una vez reciba esta señal crea el proceso pZoneManager y pasa al estado sWaitConfirmZoneManager especificando el controlador de zonas que le corresponde a éste. Como el entorno solicito la creación de un pZoneManager entonces su PID por defecto es cero, por eso el parámetro que indica el identificador del controlador de zonas tiene el mismo valor. Una vez el proceso pZoneManager haya asignado el identificador de su controlador, enviará una señal sIamZoneManager a su proceso creador el cual se encuentra en un estado permitido para recibir dicha señal y retorna al proceso que solicitó inicialmente la creación la señal sOkCreateZoneManager enviando el identificador del nuevo proceso creado. Ambos procesos terminan su interacción y quedan en el estado Idle.

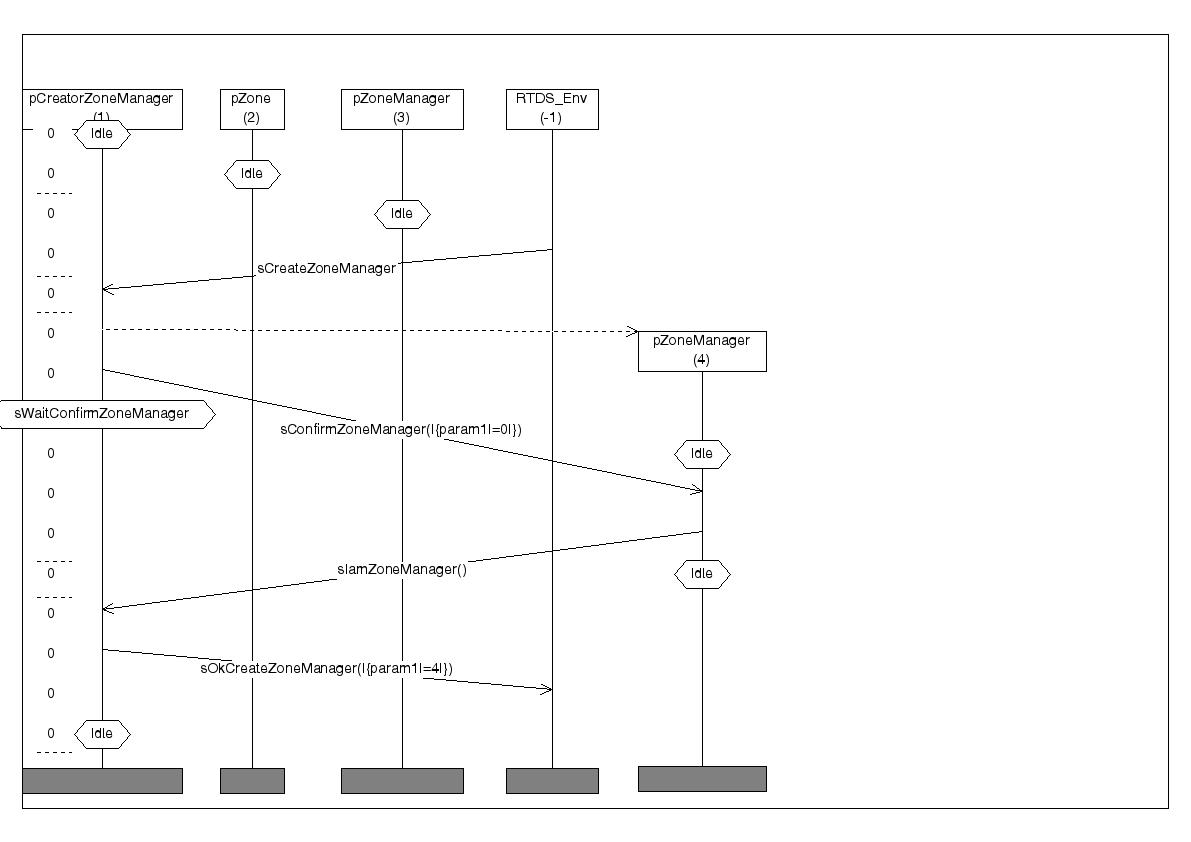


Figura 34. Resultado de la prueba de crear una instancia de un proceso pZoneManager

La tabla 5 muestra los resultados de las pruebas realizadas a distintos bloques del sistema de parqueo usando la herramienta RTDS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Número de prueba | Proposito | Pre-condición | Post-Condición | Proceso(s) donde se efectúa la prueba |
| 1. Figura 35 | Probar que el proceso pZoneManager puede crear satisfactoriamente un proceso pZone. | El sistema cuenta con una instancia de cada proceso pZone, pZoneManager y pZoneCreatorManager. | El sistema cuenta con dos instancias del proceso pZone y una instancia tanto del proceso pZoneManager como del proceso pCreatorZoneManager. | pZoneManager |
| 1. Figura 36 | Probar que el proceso pZoneManager puede instanciar un número máximo de zonas y reportar cuando se vaya a sobrepasar dicho límite. | El sistema cuenta con una instancia de cada proceso pZone, pZoneManager y pZoneCreatorManager. | El sistema cuenta con el número máximo de zonas que puede crear el proceso pZoneManager, en este caso 3, al tratar de crear más se generará una señal que no es posible crear más procesos pZone. | pZoneManager |
| 1. Figura 37 | Probar que se puede inicializar las plazas libres y totales de una zona del sistema de parqueo. | El sistema cuenta con una instancia de un proceso pZone, pZoneManager y pCreatorZoneManager. | El sistema cuenta con dos zonas, a la nueva zona se inicializa sus zonas totales y libres. | pZone y pZoneManager |
| 1. Figura 38 | Probar que la zona de parqueo es capaz de reportar una cifra correcta de plazas libres después de haber entrado un vehículo a ésta. | EL sistema cuenta con una zona de parqueo que tiene 300 plazas libres y totales. | La zona reportará que han ingresado dos vehículos por lo tanto su valor de plazas libres es de 298. | pZone |
| 1. Figura 39 | Probar que la zona reporta un número correcto de plazas libres después de ingresar y salir vehículos de dicha zona. | EL sistema cuenta con una zona de parqueo que tiene 300 plazas libres y totales. | La zona reportará que han ingresado dos vehículos y salido uno. Por lo tanto, su valor de plazas libres es de 298. | pZone |

Tabla 5. Resultado de las pruebas realizadas en el bloque BZone

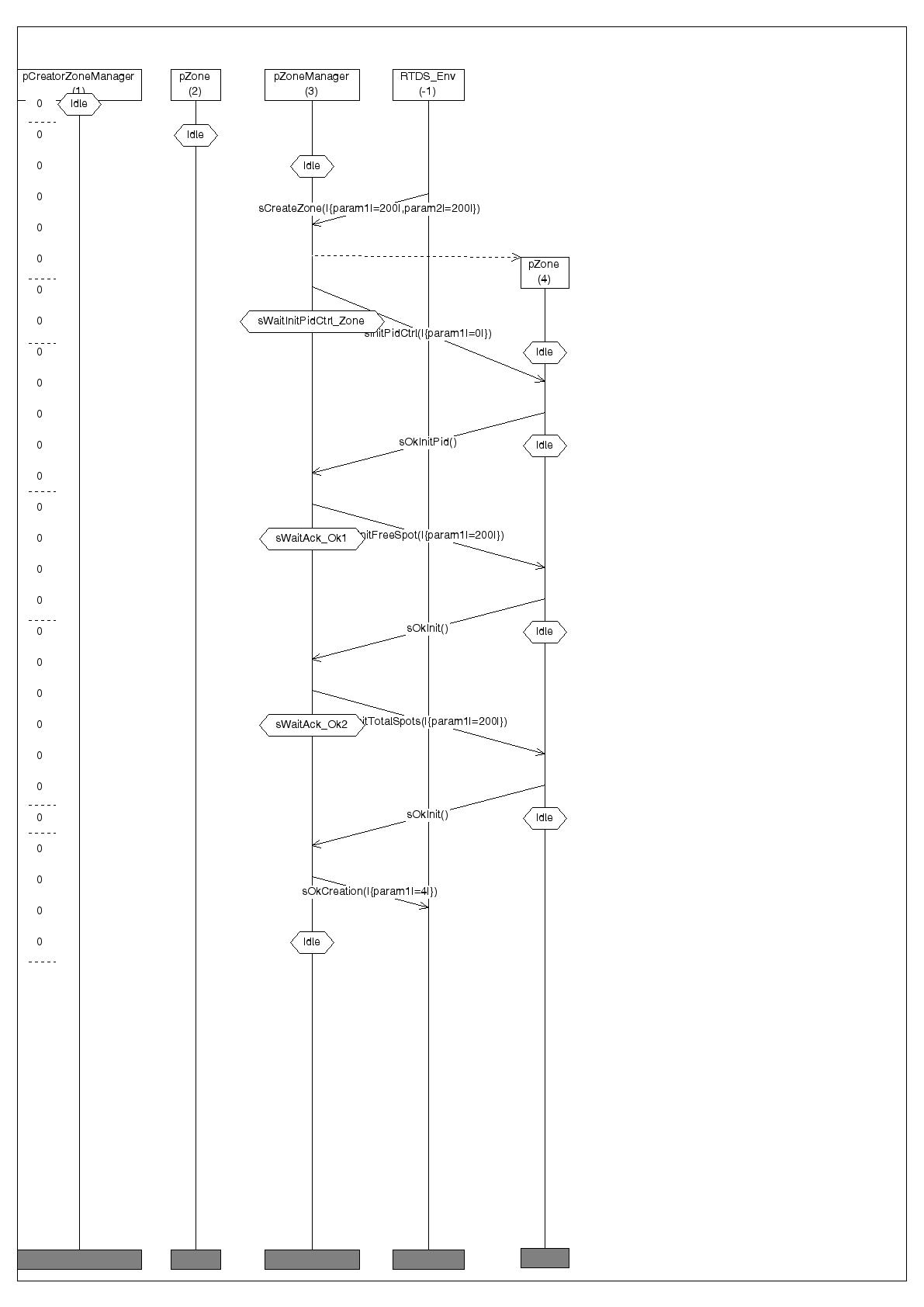


Figura 35. MSC resultado de la prueba de creación de una zona de parqueo.

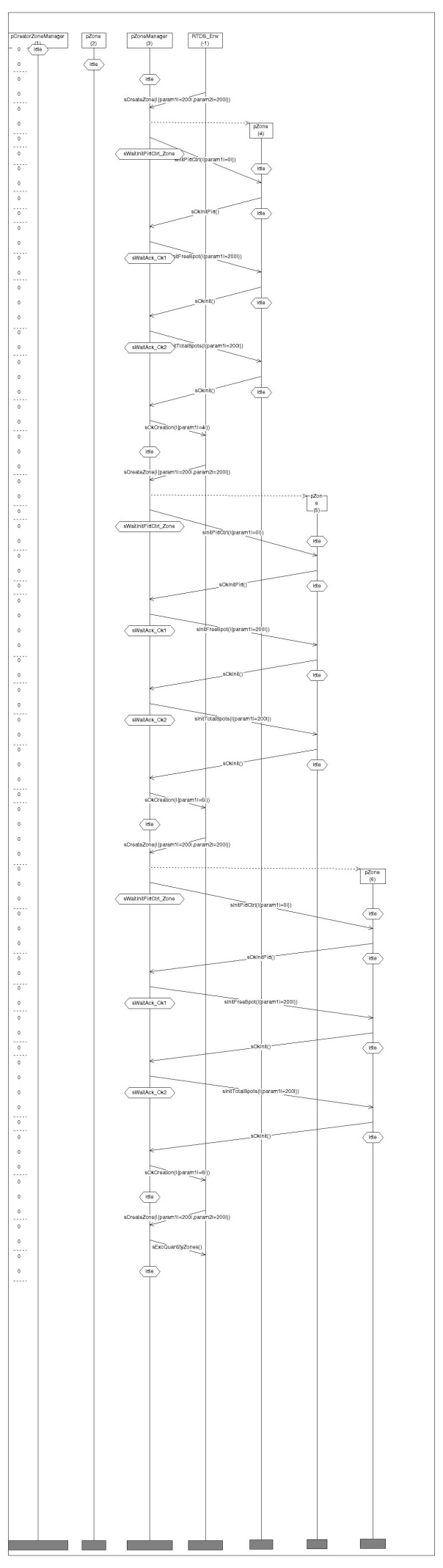


Figura 36. MSC resultado de la prueba de creación de número máximo de pruebas permitido por un proceso pZoneManager.

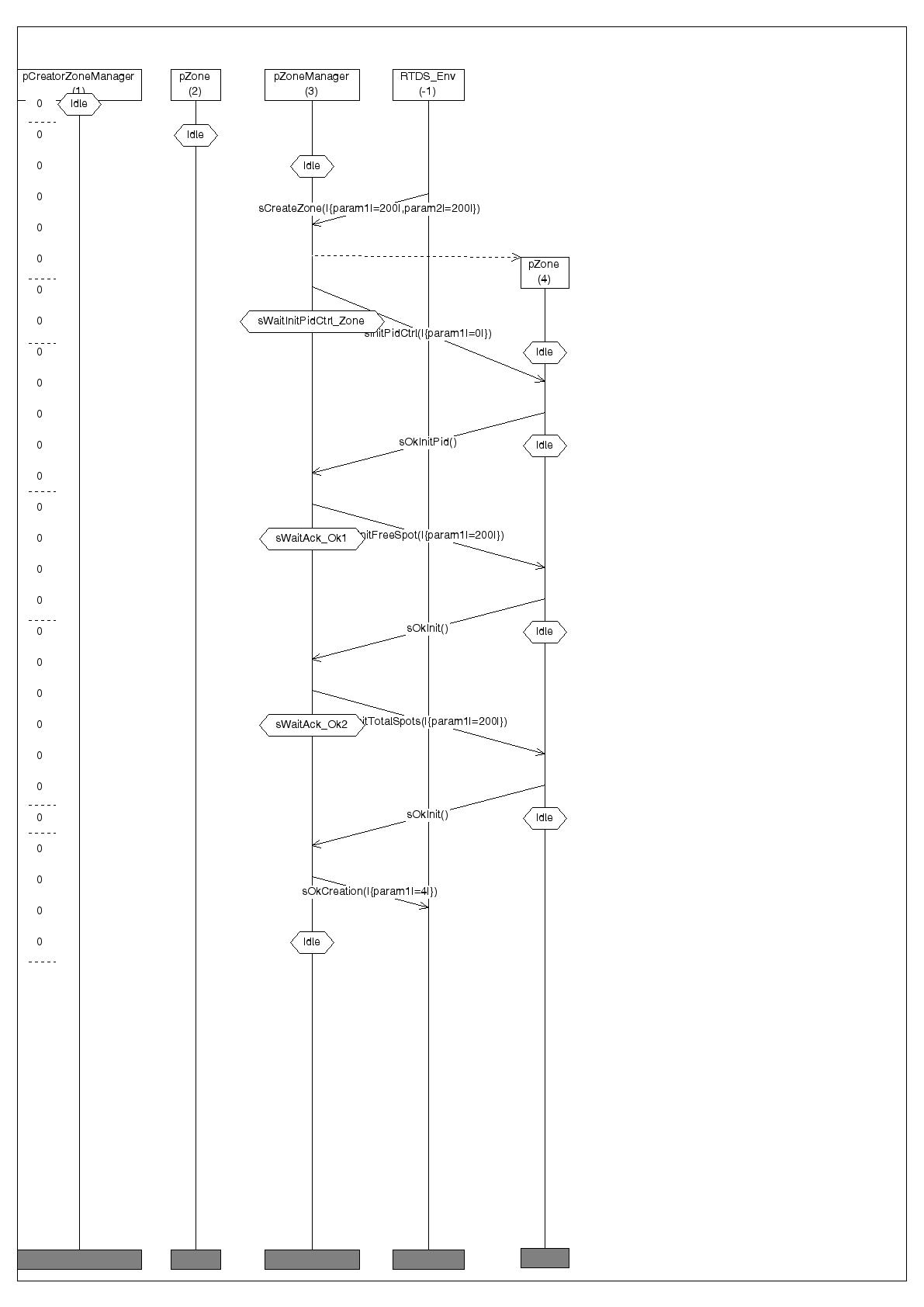


Figura 37. MSC resultado de la prueba de inicializar las plazas totales y libres en una zona de parqueo.

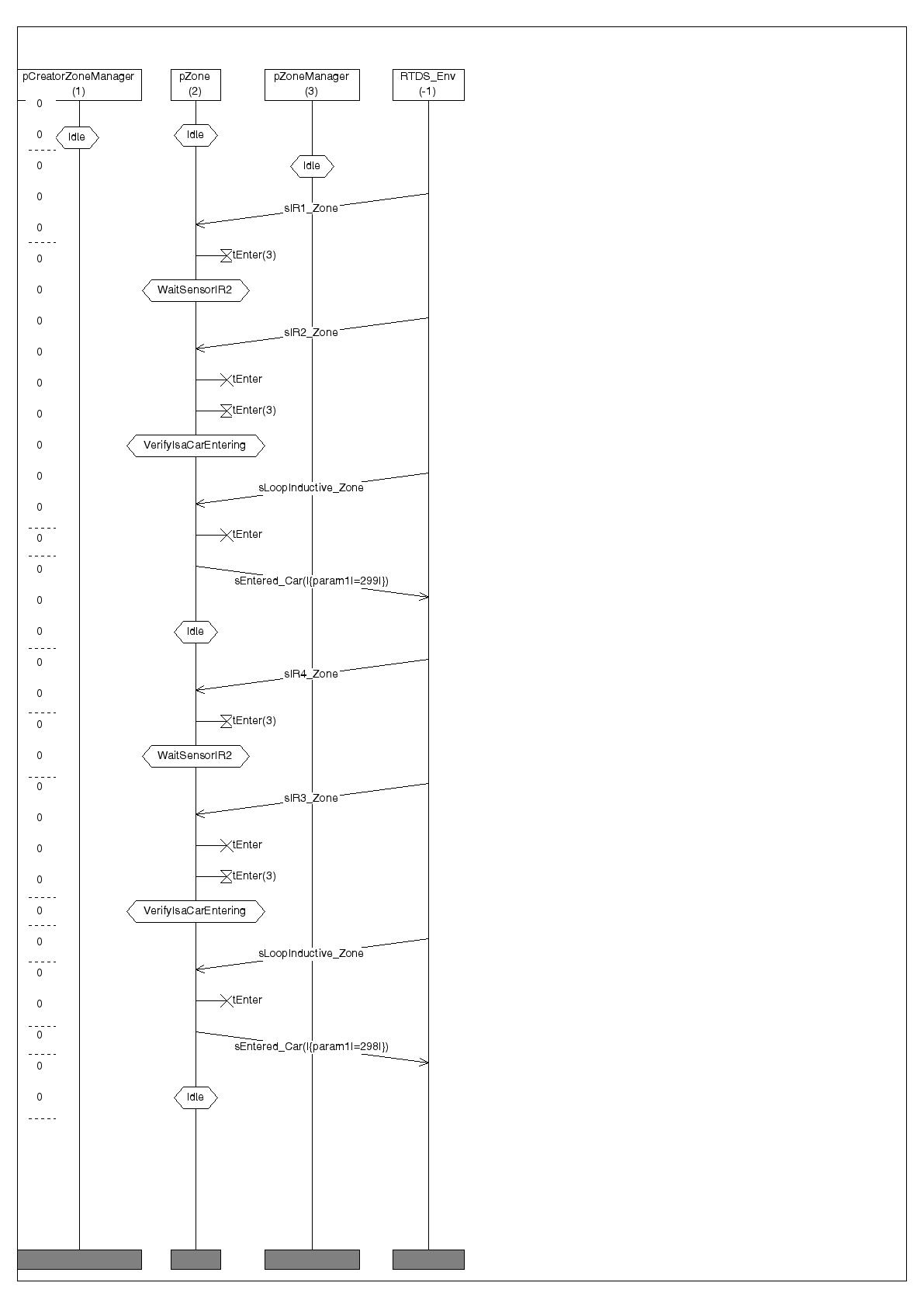


Figura 38. MSC resultado de la prueba del ingreso de un vehículo en una zona de parqueo.

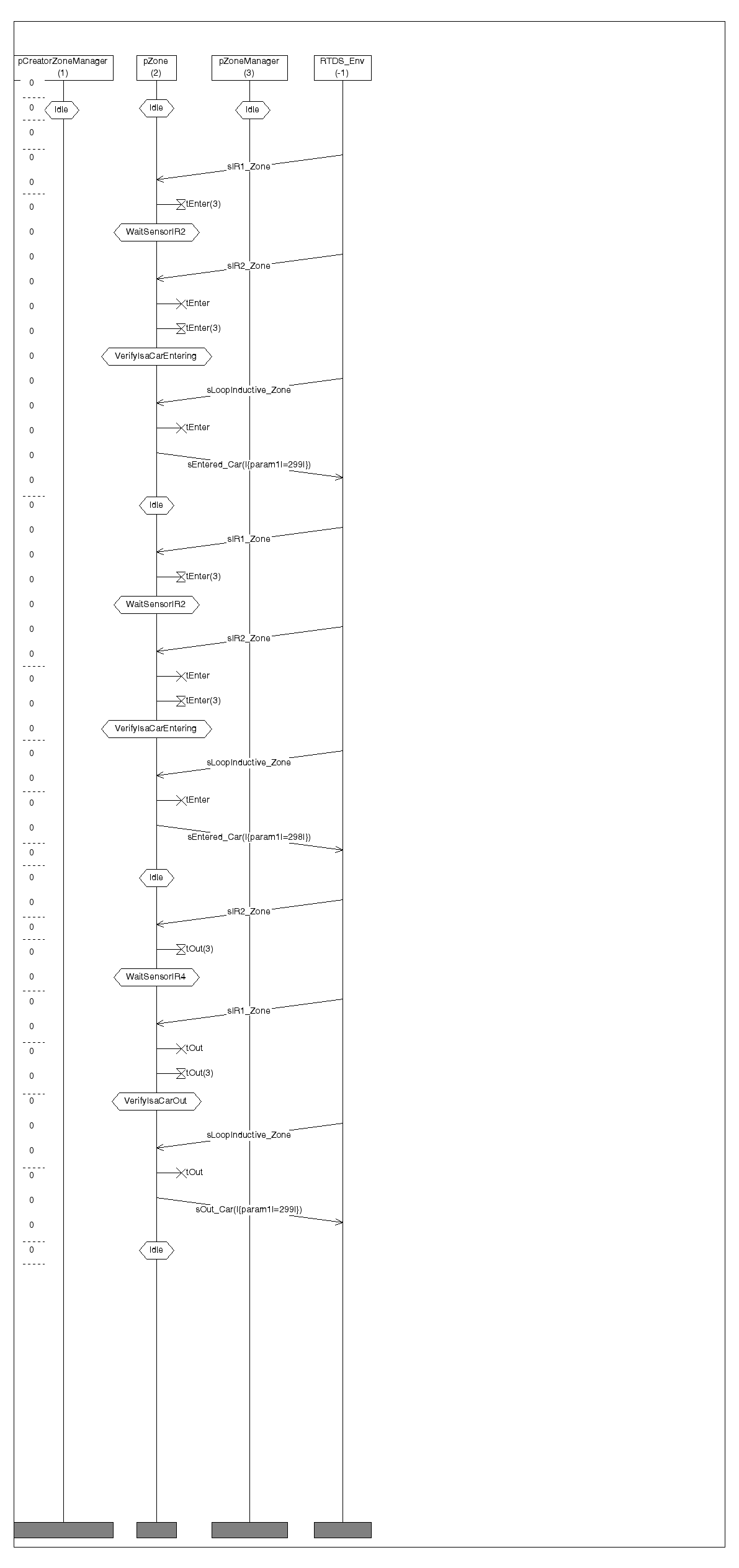


Figura 39. MSC resultado de la prueba de la salida de una vehículo de la zona de parqueo.

## Verificación Formal usando IFx

1. Una zona es un área donde se pueden aparcar vehículos y cuenta con un sistema de sensores para determinar cuando entran o salen los vehículos. [↑](#footnote-ref-1)
2. La autogestión del sistema se refiere a que el sistema de parqueo puede pedir información a los controladores de zonas de forma independiente. [↑](#footnote-ref-2)
3. El proceso pTesting solo tiene sentido en la fase de pruebas del modelo, pero no debe ser considerado en la fase de implementación. [↑](#footnote-ref-3)
4. PRAGMADEV. Reference Manual. Consultado el día 27 de Agosto del 2014 url: <http://www.pragmadev.com/downloads/RefManual.pdf> [↑](#footnote-ref-4)