# *Algoritmos y programación 3*

## *Trabajo Práctico 2*

Integrantes:

- Ursino Ian Mika, padrón: 104509
- Aguire Walter Mario, padrón: 106328
- Rivera Villatte Manuel, padrón: 106041
- Gimenez Claudio Alberto, padrón: 105509

Fecha de entrega: 01/07/2022

# Modelo

El juego consiste principalmente en mover un vehículo a través de un mapa, con el objetivo de llegar a la salida ubicada en cierta posición del mismo. A medida que el vehículo se mueve, se puede ir encontrando con entidades que interactúan de alguna forma con este al atravesarlas.

La clase principal que toma el rol de “fachada” es Juego. Todos los mensajes enviados por el usuario pasan por Juego. Es ésta quien está al tanto de qué estado tiene el juego (menú, plena partida o visualización de rankings, por ejemplo). Es quien almacena la lista de jugadores a medida que partidas se terminan, y quien delega a Jugador cuando se ejecuta alguna acción dentro del marco de una partida, como se observa en la siguiente Figura.

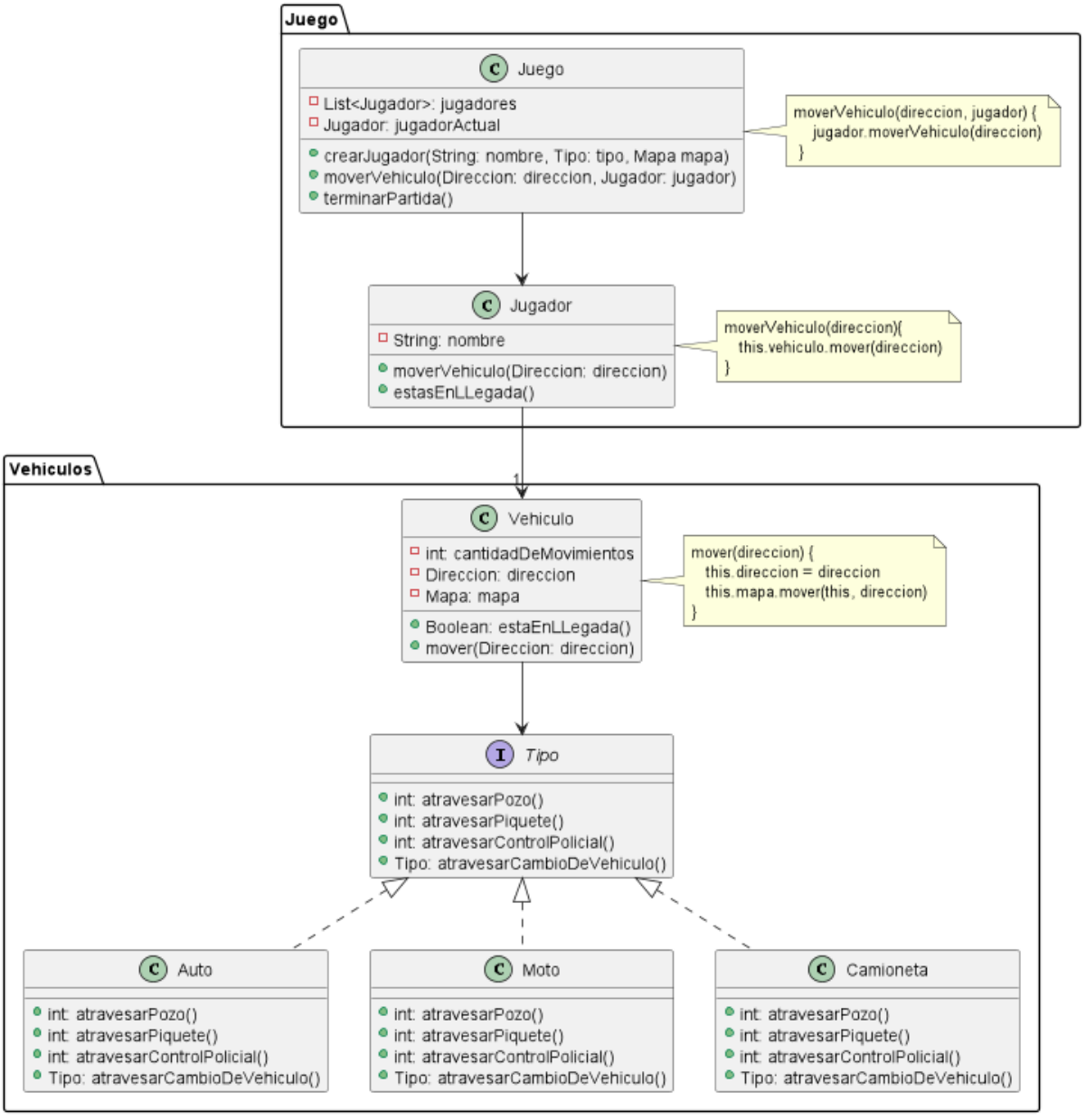


Figura 1: Modelo de Juego y Vehículos

Jugador posee un nombre como identificador, y su vehículo. Es Juego quien mantiene el registro de los Jugadores que han participado.
Para modelar al vehículo utilizamos el patrón State, el cual nos permite alterar el comportamiento del mismo frente a los objetos interactuables según cual sea su estado, que en nuestro caso lo llamamos tipo. Como se puede observar en la Figura 1, los distintos tipos de vehículo que hay son: Moto, Auto y Camioneta. Cada uno de ellos tiene un comportamiento diferente frente a las entidades que se pueden encontrar en el mapa. Por dar un ejemplo, la camioneta y el auto no pueden atravesar los piquetes, a diferencia de la moto que sí puede hacerlo.
Quien se encarga de controlar y modificar la posición del vehículo es una clase Mapa, que representa el escenario en donde se desarrolla todo el juego. Esta misma contiene las calles donde se posicionan las entidades interactuables y la posición de llegada para ganar la partida. En nuestra implementación utilizamos un Hash en donde se guardan aquellas calles que contienen elementos como pueden ser obstáculos o sorpresas que afectan de alguna manera al desarrollo de la partida. Es decir que en nuestro modelo, las calles vacías no tienen entidad propia ya que no lo necesitamos, debido a que quien se encarga de controlar si el juego termina es el mapa, y esto lo hace mediante un chequeo en el cual se verifica que la posición del vehículo y la de llegada coincidan.
Para ciertas clases necesitamos utilizar números que se generen aleatoriamente. Por ejemplo, para lograr que los interactuables se coloquen en distintas posiciones en cada partida que se inicia. Para esto creamos una clase Aleatorio que encapsula este comportamiento de generar números para las clases que lo necesiten. Las clases que utilizan Aleatorio y los métodos que tiene se muestran en la Figura 2.

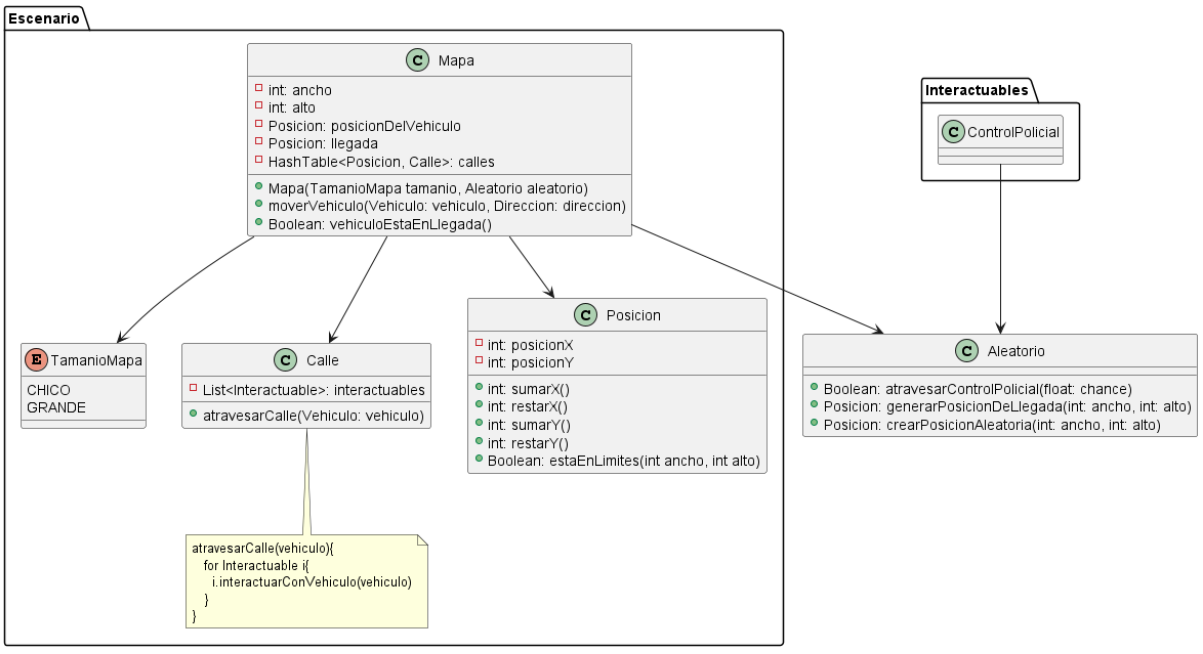


Figura 2: El Escenario

A aquellas entidades que interactúan con el vehículo, siendo estos los obstáculos y las sorpresas, las agrupamos bajo una clase Interactuable. Esto lo hicimos así ya que el comportamiento de las mencionadas anteriormente únicamente es interactuar con el vehículo del jugador, lo que nos permite reutilizar código y no tener que crear métodos

diferentes para cada tipo de Interactuable. Aún así, cada interactuable tiene la tarea de delegar al vehículo mismo, dado que es necesario saber también qué tipo de vehículo es para saber cómo comportarse frente al interactuable. Ante esta situación donde hace falta información tanto del interactuable como del vehículo, optamos por utilizar un double dispatch, donde cada tipo de vehículo tiene un método concreto por cada interactuable posible, y los interactuables delegan su comportamiento a ese método. Esta solución es mejorable, dado que si hubiera una forma polimórfica de resolver las interacciones, en el caso hipotético de que se agregue un nuevo tipo de interactuable, no sería necesario entrometer en Vehículo para añadir el método concreto que corresponda.

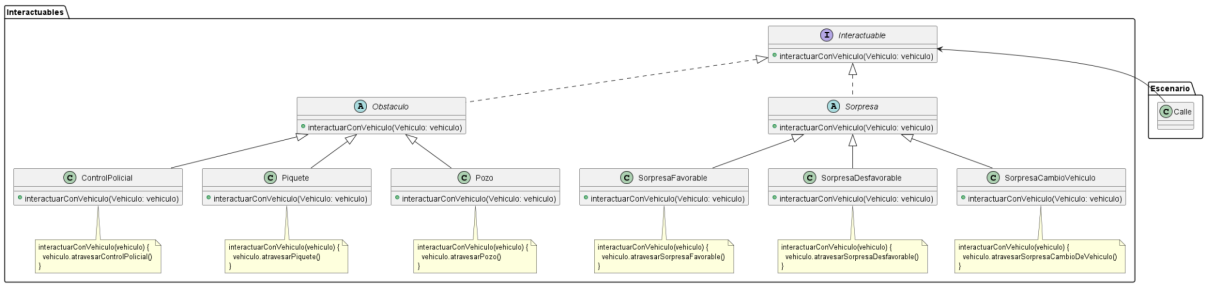


Figura 3: Diagrama de Clase de Interactuables

Con respecto al movimiento del vehículo, utilizamos una interfaz Dirección, que posee dos métodos: calcularPosicionSiguiente modifica la posición recibida en función de la dirección en donde se mueve el vehículo, y opuesto devuelve una dirección cardinalmente contraria a la misma. Por ejemplo, una instancia de la clase Arriba, ante el mensaje calcularPosicionSiguiente va a aumentar la coordenada Y de la posición que recibe, y ante el mensaje opuesto, va a devolver una instancia de la clase Abajo. A continuación, en la Figura 4, se ilustra esta asociación.

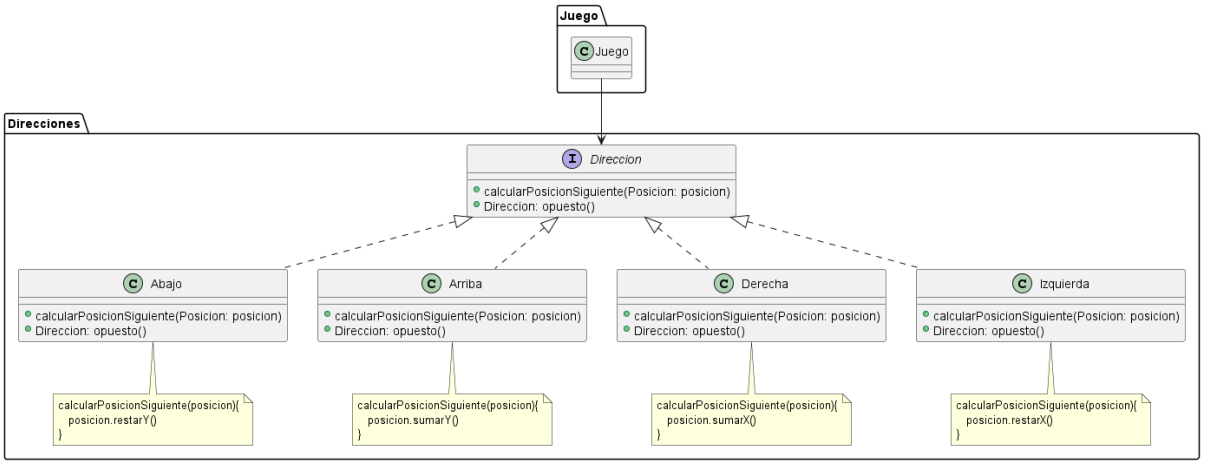


Figura 4: Diagrama de clase de Direcciones

# Interfaz

Para conectar nuestro modelo del juego con la interfaz que desarrollamos, utilizamos el patrón MVC, el cual permite separar la manera en que se desarrolla el juego de su visualización a través de la pantalla. Mediante el uso de las vistas por ejemplo, podemos mostrar distintas secciones del juego tales como el menú de inicio o el desarrollo de una partida. Para lograr una interacción entre el usuario y la vista agregamos botones necesarios que disparan eventos al ser presionados o seleccionados. Esto se hace principalmente para moverse a través del menú y para mover al vehículo a lo largo de la partida.
Los controladores por otro lado, son aquellos que actúan sobre el modelo y la vista a la vez, controlando el flujo de cambios en el primero de manera que esto se pueda ver reflejado en la vista al momento de que ocurra un cambio. Por dar un ejemplo de cómo funciona, tenemos un controlador del mapa que se encarga de mover al vehículo en la instancia del juego actual, que al momento de hacerlo notifica a las clases de la vista que están observando estos movimientos para luego mostrarlo en la interfaz gráfica.

# Uso de la aplicación

Al iniciar la aplicación, se observa el menú principal, el cual nos permite iniciar una nueva partida o ver los resultados de partidas de jugadores que ya hayan completado una anteriormente.
Al clickear sobre el botón para iniciar el juego, nos pide ingresar un nombre con el cual se nos podrá identificar en el ranking. Además de esto, se permite seleccionar con que vehículo se empieza el juego y la dificultad del mismo.
Una vez comenzada la partida, se debe ir moviendo el vehículo hasta llegar a la posición marcada con una bandera, que indica la posición de llegada para terminar.
El mapa se ve oculto completamente, salvo por una posición alrededor del vehículo que se irá modificando a medida que el mismo se mueva.
Para mover al vehículo, basta con clickear sobre los botones que indican la dirección deseada. Y con respecto a los objetos interactuables hay que hacer lo mismo: Si se quiere pasar sobre una sorpresa simplemente hay que mover el vehículo a través de la misma. Esto no será posible hacerlo sobre los piquetes (exceptuando el caso de que el vehículo sea una moto), por lo que habrá que encontrar un camino alternativo si nos topamos con uno de estos.
Una vez que posicionamos el vehículo en la posición final, se observa el puntaje obtenido, y se presenta la opción de volver al menú principal o de ver el ranking de las partidas.

# Diagramas de Secuencia

A continuación se presenta(n) diagrama(s) de secuencia realizados para ilustrar interacciones complejas entre las entidades del modelo, en un caso de uso genérico. Para mayor visibilidad, se presentan de forma horizontal.

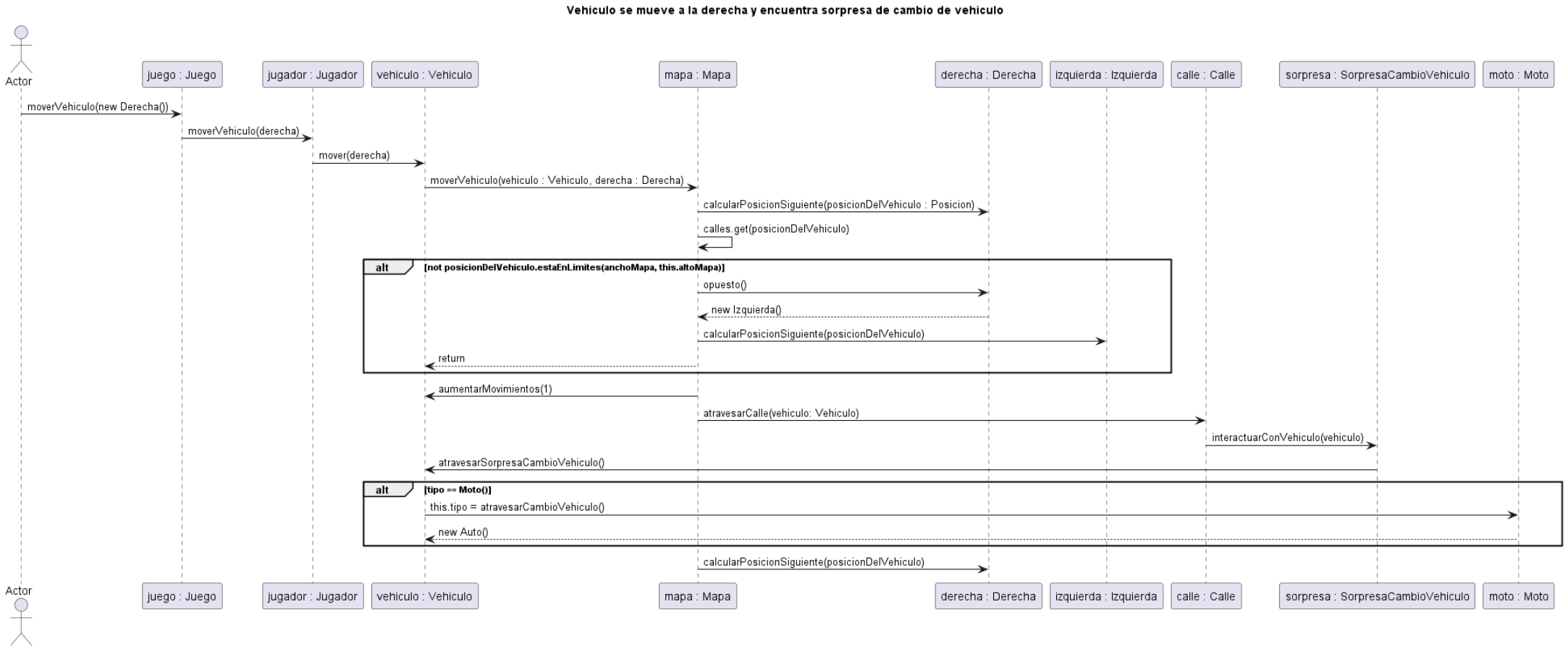


Figura 1: Moto atraviesa sorpresa de cambio de vehiculo, se convierte en auto

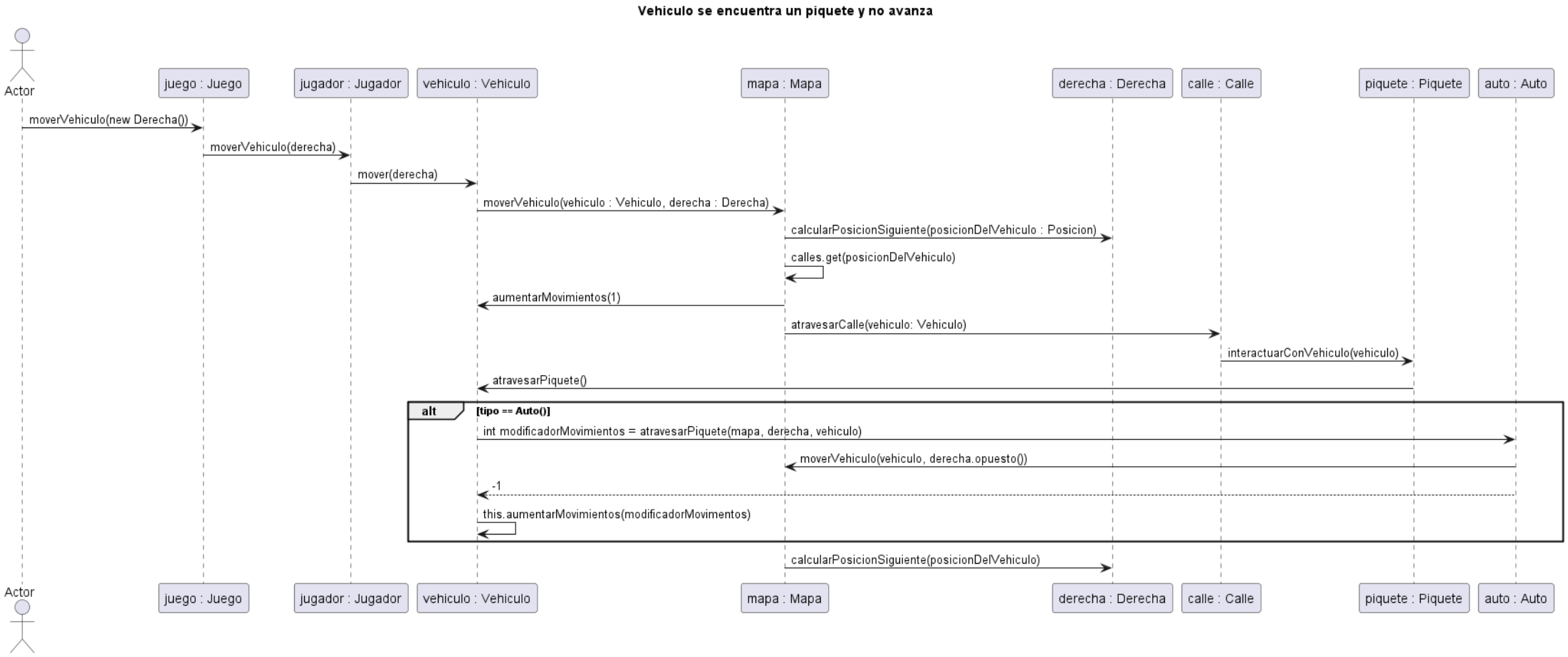


Figura 2: Un auto se encuentra un piquete y no puede avanzar