Tecnológico de Monterrey

Actividad Integradora

Modelación de Sistemas Multiagentes y Gráficas Computacionales (Gpo. 523)

Héctor Arturo Noyola Mondragón

A01023808

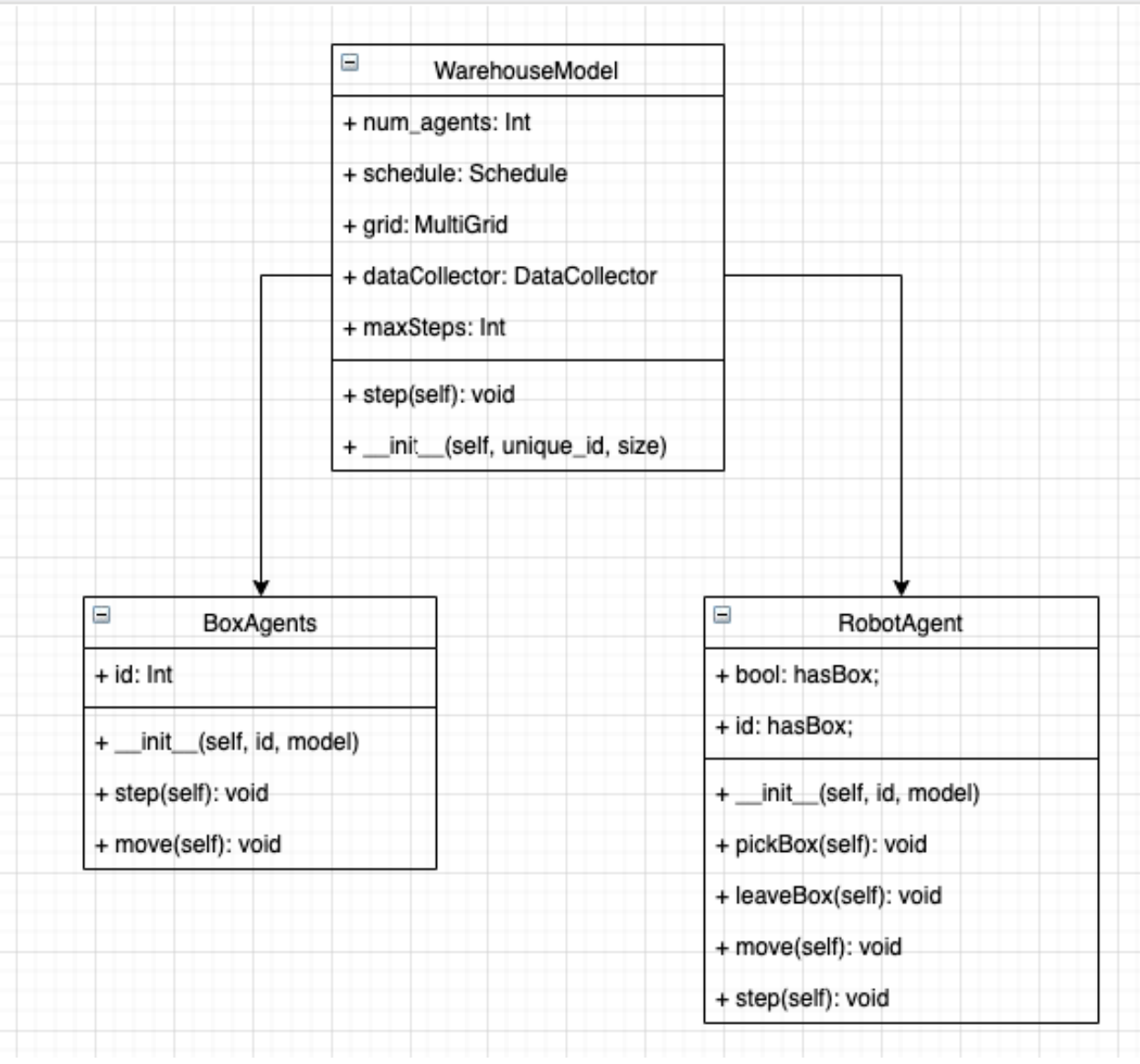
Profesor:

Christopher David Balderas Silva

Sergio Ruiz Loza

Nuestra problemática nos plantea un escenario donde tenemos un almacén con cajas desordenadas que deben ser acomodadas. A nuestra disposición hay cinco robots que deberán apilar estas cajas en máximo cinco por celda. Los robots podrán: saber cuándo están cargando una caja, saber si a su alrededor hay una caja, una pared u otro robot.

Para resolver este problema, se tiene una solución basada en agentes. Pensando en un modelo basado en la librería mesa, tenemos el siguiente diagrama de clases.



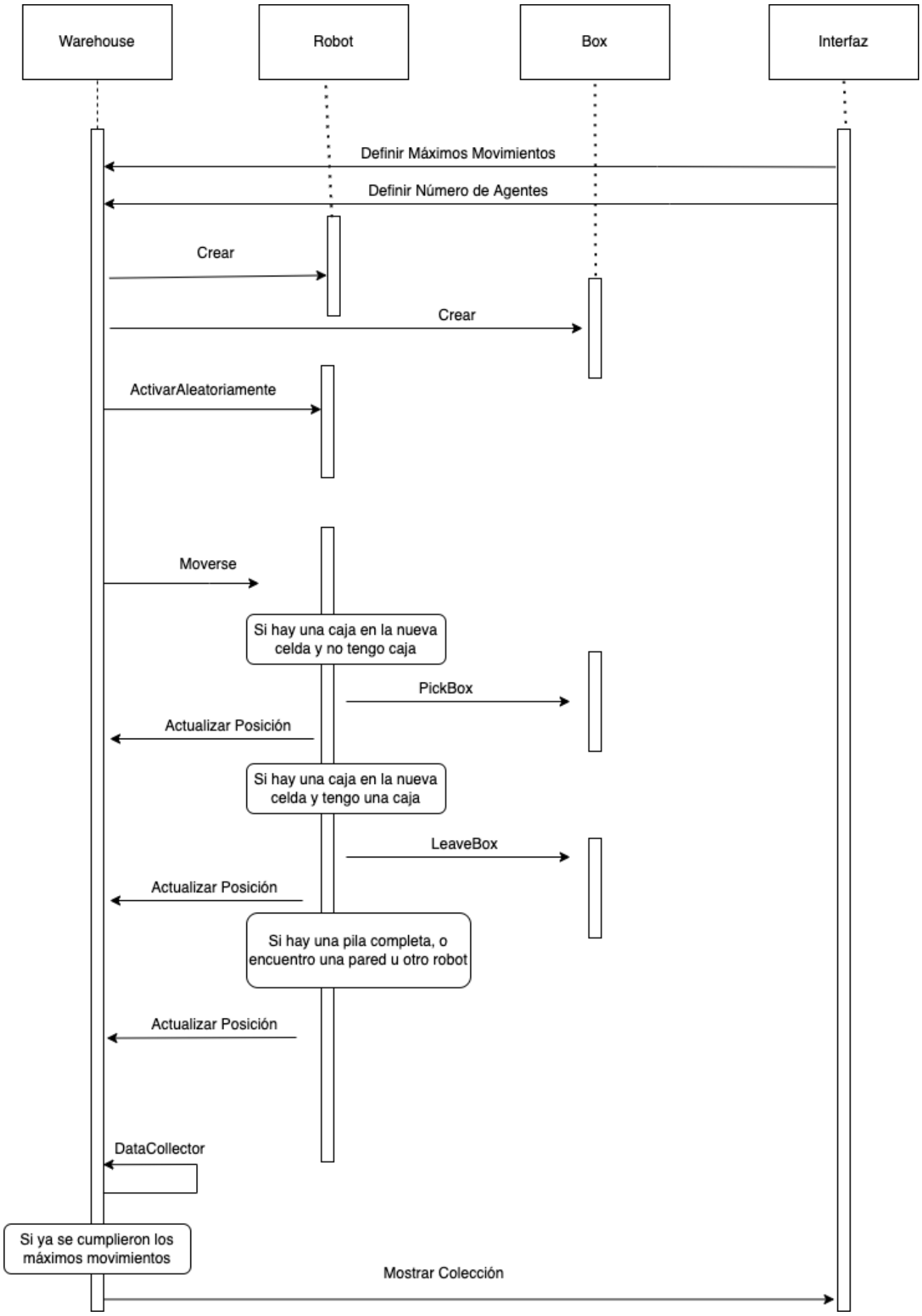
La clase “WarehouseModel” funcionará como el ambiente de nuestra solución. Este tendrá un número de agentes, 5 robots y una cantidad X de cajas que se deben apilar. Para simular el almacén, utilizamos una clase predeterminada “MultiGrid”. Este tipo de grid permite que múltiples agentes existan en la misma celda, de esta manera podemos apilar una caja sobre otra en el ambiente. La última variable que no pertenece a la clase Model y le tenemos que declarar externamente es el número máximo de pasos que se darán en este ambiente. Adicionalmente, hay dos atributos que vienen incluidos en la librería de Mesa: schedule y dataCollector. La variable “schedule” es responsable de hacer que los agentes den un paso cuando el ambiente así lo requiera. Este schedule es de tipo RandomActivation, es decir, activará a los agentes del ambiente uno por uno en orden aleatorio. Finalmente, dataCollector almacenará los datos que sean necesarios durante cada paso: movimientos totales de los robots, y tiempo necesario para lograr que las cajas queden apiladas. Como parte de sus funciones, tenemos la función “step” que llamará al scheduler para avanzar un paso, y que hará que el dataCollector recopile los datos necesarios. Por último, su __init__ recibe la cantidad de celdas que tendrá la cuadrícula, un número N de agentes y su id.

La clase “RobotAgent” es la encargada de simular a los robots en el almacén. Tiene un valor booleano “hasBox”, el cual será importante para determinar si una caja necesita ser puesta en una pila o si deba tomarla para buscar una posición donde la pueda poner. Como funciones tiene __init__ que recibe su id y el modelo al que pertenece. La función pickBox toma una caja y la mantiene en la misma posición del robot, de esta manera se van moviendo de la misma manera hasta volar a ser apilada con la función leaveBox. Ésta se encarga de dejar una de las cajas en la posición de la cuadrícula apropiada, el modelo se encarga de aumentar el número de agentes en el grid. La función step se encargará de llamar al resto de las funciones si es apropiado. Siempre se llamará la función de Move, la cual buscará las posiciones de sus vecinos. Con base en sus resultados, hará un movimiento y determinará si es necesario utilizar pickBox o leaveBox

La clase “BoxAgent” funciona de modo similar a RobotAgent con menos atributos. Únicamente se moverá y tendrá un id. La función step será encargada de

llamar a move, si un agente con hasBox está en su misma posición, entonces tendrá su mismo movimiento.

Lo primero que se hace es definir el número de agentes y el número máximo de movimientos que nuestro modelo podrá correr. Una vez determinados esos números, se crearán los 5 robots y el número de las cajas. Cuando se hace el primer paso del scheduler, se escoge uno de los agentes de manera aleatoria, en caso de ser un Robot, está se moverá la casilla apropiada y tendrá tres posibilidades según lo que encuentre en esta celda. Si hay una caja y no carga nada, la toma. Si tiene una caja y encuentra otra caja, la apila. Si encuentra una pila completa, una pared, u otro robot, simplemente actualiza su posición en la cuadrícula del modelo. Se repite este proceso en el número de movimientos establecido. Después de cada Step, data collector recopilará los datos necesarios que deberán ser mostrados en la interfaz al finalizar el modelo.


Warehouse
Robot
Box
Interfaz
Definir Máximos Movimientos
Definir Número de Agentes
Crear
Crear
ActivarAleatoriamente
Moverse
Si hay una caja en la nueva celda y no tengo caja
PickBox
Actualizar Posición
Si hay una caja en la nueva celda y tengo una caja
LeaveBox
Actualizar Posición
Si hay una pila completa, o encuentro una pared u otro robot
Actualizar Posición
DataCollector
Si ya se cumplieron los máximos movimientos
Mostrar Colección