MEMORIA

Sistema Domótico para el Cuidado de Personas: Gestión y Administración de Medicamentos

Óscar Marqués Juan

Alejandro López Gómez

Objetivos

Objetivo principal

Desarrollar un sistema domótico multiagente para la gestión y administración de medicamentos en entornos domésticos.

Objetivos específicos

- Implementar un sistema donde el agente owner pueda tomar medicaciones pautadas (al menos 5 medicamentos con pautas específicas).
- Desarrollar la capacidad de movilidad libre para el owner por toda la casa y permitir que descanse en zonas adecuadas (silla, sofá, cama).
- Crear un sistema donde la medicación pueda ser proporcionada tanto por el agente robot como por el propio owner.
- Implementar la comunicación de pautas del owner al robot al inicio del sistema.
- Desarrollar un mecanismo para que el owner informe al robot cuando ha tomado la medicación.
- Implementar la funcionalidad para que el owner modifique sus pautas de medicación con el tiempo.
- Crear zonas de almacenamiento de medicamentos accesibles para ambos agentes.
- Mantener un sistema de inventario que garantice la disponibilidad de medicamentos.
- Desarrollar la navegación del robot para moverse libremente por la casa sorteando obstáculos y localizando al owner.
- Implementar un sistema de verificación para que el robot compruebe el consumo efectivo de la medicación.

Resumen de la solución propuesta

Hemos creado un sistema inteligente para el hogar que usa múltiples agentes virtuales para simular una casa donde un robot asistente colabora con el dueño en el proceso de administración del medicamento. Este sistema está construido con la ayuda de Jason, un intérprete del lenguaje AgentSpeak, que nos proporciona la posibilidad de desarrollar agentes que operan según el modelo BDI .

1. **Agente Owner**: Simula a una persona con necesidades de medicación que se desplaza por la casa y realiza actividades cotidianas como descansar en el sofá o dormir en la cama. Este agente tiene la capacidad de tomar medicamentos por iniciativa propia, solicitar asistencia al robot y modificar sus pautas de medicación. Inicialmente, transmite al robot sus pautas de medicación.

2. **Agente Robot**: Actúa como asistente, monitorizando las pautas de medicación del owner, entregando los medicamentos en los horarios programados y verificando su consumo efectivo. Este agente se desplaza por la casa buscando al owner cuando es necesario, evitando obstáculos y otros agentes.

El entorno de la simulación incluye:

- Diversas habitaciones que están conectadas entre ellas.
- Muebles y objetos como sofás, camas …
- Una zona específica para guardar los medicamentos.

Para la navegación de los agentes hemos implementado el algoritmo AStar, para poder encontrar las rutas óptimas a seguir en el entorno, evitando los obstáculos y evitando entre ellos.

La gestión de medicamentos incluye:

- Un inventario inicial con diferentes tipos de medicamentos
- Pautas de administración definidas con nombre, hora y frecuencia
- Un sistema de reposición automática cuando los medicamentos se agotan
- Visualización gráfica de la apertura de gabinetes y la entrega de medicamentos

El paso del tiempo se simula mediante un componente Calendar que permite controlar la velocidad de la simulación y observar comportamientos específicos en diferentes momentos del día.

Planificación y seguimiento

A continuación, se detalla la planificación y ejecución de cada fase, con énfasis en los requisitos semanales establecidos.

Planificación inicial

La planificación del proyecto se estructuró en tres fases principales, correspondientes a las semanas 5, 6 y 7 del cronograma del curso:

Detalle del desarrollo por semanas

Semana 5: Adaptación del entorno y sistema de navegación básico

1. **Modificación del desplazamiento de agentes**: Se implementó un sistema de movimiento por casillas individuales (izquierda, derecha, arriba, abajo), sustituyendo el movimiento continuo original

2. **Zonas de almacenamiento de medicamentos**: Se diseñó e incorporó una zona principal de almacenamiento: un gabinete en la cocina para medicamentos. Estos elementos se definieron en la clase HouseModel.java y se visualizaron mediante HouseView.java.

3. **Posicionamiento de agentes**: Se desarrolló la funcionalidad para que ambos agentes pudieran posicionarse junto a las zonas de almacenamiento y uno al lado del otro. Esto implicó la creación de acciones específicas en los archivos robot.asl y owner.asl para buscar y ocupar posiciones adyacentes a ubicaciones objetivo.

Semana 6: Sistema de navegación avanzado y comunicación entre agentes

1. **Desplazamiento por toda la casa**: Se perfeccionó el sistema de navegación para permitir que tanto el owner como el robot pudieran desplazarse libremente por todas las habitaciones. Para que nuestros agentes puedan moverse libremente por todos lados en el espacio, usamos un algoritmo llamado A*. Este algoritmo funciona así:

   Crea una cuadrícula que representa toda nuestra casa.
   Marca qué cuadros están bloqueados (como paredes o muebles) donde el agente no puede pasar.
   Permite que el agente se mueva tanto en línea recta como en diagonal.

Utiliza cálculos especiales (heurísticos) para encontrar el camino más eficiente de un punto a otro.
Este sistema es como si tuviéramos un mapa cuadriculado de la casa, donde el algoritmo calcula automáticamente la mejor ruta para que el agente llegue a su destino, evitando los obstáculos.
Se implementó la detección de obstáculos para evitar colisiones durante el desplazamiento.

2. **Comunicación de pautas de medicación**: Se implementó el sistema de comunicación entre el owner y el robot para la transmisión de información sobre medicamentos. El owner, al inicio de la simulación, envía al robot un listado completo de sus medicamentos con sus respectivas pautas de administración (nombre, horario y frecuencia). Esta comunicación se realiza mediante mensajes entre agentes utilizando el sistema de creencias de Jason.

3. **Transformación del owner**: Se convirtió al owner de un elemento pasivo del entorno a un agente activo con comportamientos propios. Se le dotó de la capacidad para desplazarse autónomamente, descansar en diferentes lugares de la casa y tomar decisiones sobre su medicación. Se implementaron patrones de comportamiento que simulan actividades cotidianas como sentarse en el sofá, acostarse en la cama o ir al baño.

Semana 7: Comportamientos específicos y verificación

1. **Acceso aleatorio a medicamentos**: Se desarrolló la funcionalidad para que el owner accediera aleatoriamente al armario de medicinas y tomara la medicación según las pautas establecidas. Se implementaron comportamientos probabilísticos que determinan si el owner decide tomar su medicación por iniciativa propia o esperar a que el robot se la entregue. Se incluyó la gestión de posibles colisiones en las horas pautadas mediante un sistema de prioridades.

2. **Sistema de notificación**: Se implementó el mecanismo para que el owner notificara al robot cuando había tomado una medicación. Esta notificación se realiza mediante mensajes entre agentes, incluyendo la información del medicamento tomado y la hora de administración. El sistema gestiona adecuadamente las situaciones donde varias medicaciones coinciden en el mismo horario.

3. **Verificación de consumo**: Se desarrolló la funcionalidad para que el robot verificara el consumo efectivo de la medicación. Esto incluye la comprobación del inventario de medicamentos y el seguimiento de las notificaciones del owner. El robot mantiene un registro actualizado de las medicaciones tomadas y pendientes.

4. **Sistema de entrega de medicamentos**: Se implementó el mecanismo completo para que el robot entregara los medicamentos al owner según las pautas programadas. Esto incluye la búsqueda del owner por la casa, el posicionamiento adecuado para la entrega, la animación de la entrega y la actualización del inventario y registros.

Arquitectura

El sistema desarrollado se basa en una arquitectura multiagente que integra varios componentes para simular un entorno doméstico inteligente centrado en la gestión de medicamentos. A continuación, se describe la arquitectura general del sistema,

Arquitectura general

La arquitectura del sistema sigue un modelo de capas, con una clara separación entre:

1. **Capa de agentes**: Implementa la lógica de comportamiento de los agentes (robot y owner)
2. **Capa de entorno**: Gestiona el estado del mundo y las percepciones
3. **Capa de visualización**: Proporciona la representación gráfica del sistema
4. **Capa de servicios**: Implementa funcionalidades compartidas (navegación, gestión del tiempo)

Esta estructura modular facilita la modificación y extensión del sistema, permitiendo cambiar componentes específicos sin afectar al funcionamiento general.

Componentes principales

Agentes

El sistema incluye dos agentes principales, implementados mediante programas en AgentSpeak (archivos .asl):

- **Agent Robot (robot.asl)**: Implementa el comportamiento del robot asistente, incluyendo:
    - Gestión del inventario de medicamentos
    - Sistema de navegación para moverse por la casa
    - Lógica para entregar medicamentos según horarios
    - Actualización de pautas de medicación
    - Verificación del consumo de medicamentos
    - Reposición automática de stock
    - Comportamientos de patrulla cuando está inactivo
- **Agent Owner (owner.asl)**: Implementa el comportamiento del propietario, incluyendo:
    - Gestión de pautas de medicamentos (nombre, hora, frecuencia)
    - Simulación de comportamientos cotidianos (moverse, sentarse, dormir)
    - Toma de medicinas (autónoma o asistida)
    - Modificación de horarios de medicación
    - Comportamientos nocturnos específicos

- Sistema de navegación por la casa

Entorno

El entorno del sistema está implementado mediante las siguientes clases Java:

- **HouseEnv.java**: Actúa como intermediario entre los agentes y el modelo del mundo. Proporciona percepciones a los agentes, procesa sus acciones y actualiza el estado del mundo en consecuencia. También gestiona eventos temporales y coordina la interacción entre componentes.

- **HouseModel.java**: Define la estructura física de la casa, incluyendo habitaciones, muebles y objetos. Implementa la lógica para acciones como abrir gabinetes, gestionar el inventario de medicamentos y determinar si los agentes pueden moverse a determinadas posiciones.

Visualización

La visualización del sistema se implementa mediante la clase:

- **HouseView.java**: Proporciona una representación gráfica del entorno doméstico, incluyendo la casa, los muebles, los agentes y las interacciones. Implementa animaciones para acciones como la entrega de medicamentos y efectos visuales para resaltar elementos importantes.

Servicios

El sistema incluye servicios compartidos implementados en:

- **AStar.java:** Este archivo implementa el algoritmo de búsqueda A* para encontrar rutas óptimas en la casa. Así, permite a los agentes navegar por la casa de manera efectiva para evitar obstáculos y otros agentes.
- **Calendar.java:** Este otro archivo mantiene una escala de deslizamiento para simular el tiempo. Luego, permite el aumento y disminución del tiempo del sistema, así como su visualización, notificando a los componentes registrados sobre cualquier cambio en el tiempo.

Tecnologías e integración de productos de terceros

El desarrollo del sistema domótico para la gestión de medicamentos ha requerido la integración de diversas tecnologías y productos de terceros. A continuación, se describen las principales tecnologías utilizadas y detallando su integración en el proyecto.

Entorno de desarrollo y lenguajes de programación

Java

Java se ha utilizado como lenguaje principal para la implementación del entorno, la visualización y los servicios del sistema.

AgentSpeak y Jason

AgentSpeak y su implementación Jason se han utilizado para programar los agentes del sistema. Jason proporciona un intérprete para AgentSpeak y una plataforma para la ejecución de sistemas multiagente.

Bibliotecas y frameworks

Java Swing

Java Swing se ha utilizado para implementar la interfaz gráfica del sistema, incluyendo la visualización de la casa, los agentes y los controles de la simulación.

Librería de Jason para entornos

Se ha utilizado la librería estándar de Jason para la implementación de entornos personalizados. Esta librería proporciona:

- **Infraestructura para entornos**: Clases base para implementar entornos personalizados
- **Gestión de percepciones**: Mecanismos para proporcionar información a los agentes
- **Procesamiento de acciones**: Funcionalidades para procesar las acciones de los agentes
- **Integración con la plataforma**: Compatible con el ciclo de razonamiento de Jason

Herramientas de desarrollo

Visual Studio Code

Visual Studio Code se ha utilizado como entorno de desarrollo integrado para el proyecto. La elección de Visual Studio Code se justifica por:

- **Soporte para Java**:Viene con herramientas específicas para el desarrollo en Java.

- **Integración con Jason**: Existen plugins que facilitan el desarrollo de sistemas Jason
- **Depuración:** Viene con herramientas avanzadas de depuración de código.
- **Gestión de proyectos:** Ayuda en la organización y navegación del código fuente.

Git

Git se ha utilizado como sistema de control de versiones para el proyecto. La elección de Git se fundamenta en:

- **Control de cambios**: Permite registrar y gestionar modificaciones en el código
- **Trabajo colaborativo**: Facilita la colaboración entre varios desarrolladores
- **Gestión de ramas**: Posibilita el desarrollo paralelo de diferentes funcionalidades
- **Integración con plataformas**: Compatible con servicios de alojamiento como GitHub

Diseño del software

Diseño estático

El diseño estático del sistema se organiza en torno a cinco clases principales que implementan la estructura y funcionalidad del entorno domótico, así como dos agentes que proporcionan el comportamiento inteligente del sistema.

Clases principales

AStar.java Esta clase implementa el algoritmo A* para la navegación de los agentes en el entorno. Sus principales componentes son:

- Estructura de nodo para representar posiciones en el entorno
- Métodos para calcular el camino óptimo entre dos puntos
- Gestión de listas abiertas y cerradas para la exploración del espacio
- Detección y manejo de obstáculos

Calendar.java Esta clase proporciona la simulación temporal del sistema. Sus componentes incluyen:

- Control deslizante para ajustar la hora del sistema
- Temporizador para actualizar el tiempo simulado
- Sistema de notificación para informar a los observadores sobre cambios temporales
- Visualización de la hora actual
- Controles para pausar y reanudar la simulación

HouseEnv.java Esta clase conecta el entorno con los agentes y la interfaz. Sus principales componentes son:

- Métodos para procesar las acciones de los agentes
- Sistema de generación de percepciones
- Gestión del estado del mundo
- Actualización de posiciones de agentes y objetos
- Manejo de eventos temporales
- Coordinación entre componentes del sistema

HouseView.java Esta clase proporciona la visualización del entorno domótico. Sus componentes incluyen:

- Representación gráfica de la casa y sus elementos
- Métodos para dibujar agentes, muebles y habitaciones
- Animaciones para visualizar interacciones
- Efectos visuales para destacar elementos importantes
- Manejo de colores y estilos visuales

HouseModel.java Esta clase define la estructura física del entorno y la lógica de interacción. Sus componentes son:

- Definición de habitaciones, muebles y objetos
- Gestión del inventario de medicamentos
- Lógica para acciones como abrir gabinetes
- Control de estado de objetos
- Determinación de posiciones válidas para agentes
- Gestión de listas de medicamentos

Agentes

robot.asl Este archivo implementa el comportamiento del agente robot mediante planes y reglas en AgentSpeak. Sus principales componentes son:

- Creencias sobre el inventario de medicamentos
- Planes para la navegación por la casa
- Comportamiento para la entrega de medicamentos
- Lógica para actualización de pautas
- Verificación de consumo de medicamentos
- Reposición automática de stock
- Comportamiento de patrulla en momentos de inactividad

owner.asl Este archivo implementa el comportamiento del agente propietario. Sus principales componentes son:

- Creencias sobre pautas de medicamentos
- Planes para simular comportamientos cotidianos
- Lógica para tomar medicamentos
- Comportamiento para comunicarse con el robot

- Modificación de horarios de medicación
- Comportamientos específicos según la hora del día

Diseño dinámico

El diseño dinámico del sistema describe cómo interactúan los componentes durante la ejecución. A continuación, se presentan los principales flujos de interacción

Flujo de entrega de medicamentos

1. Calendar genera un evento de tiempo que alcanza una hora de medicación
2. HouseEnv notifica a los agentes sobre el cambio de hora
3. El robot detecta que es hora de una medicación
4. El robot solicita a HouseEnv la posición del gabinete de medicamentos
5. El robot utiliza AStar para calcular una ruta hacia el gabinete
6. HouseEnv procesa las acciones de movimiento del robot y actualiza su posición
7. El robot solicita abrir el gabinete
8. HouseView visualiza la apertura del gabinete
9. El robot toma el medicamento, reduciendo el inventario en HouseModel
10. El robot solicita a HouseEnv la posición del owner
11. El robot calcula una ruta hacia el owner usando AStar
12. HouseEnv procesa las acciones de movimiento y posiciona al robot junto al owner
13. El robot entrega el medicamento al owner
14. HouseView visualiza la entrega
15. El robot actualiza sus creencias sobre medicamentos entregados

Flujo de toma autónoma de medicación

1. Calendar genera un evento de tiempo que alcanza una hora de medicación
2. El owner decide tomar su medicación por iniciativa propia
3. El owner solicita a HouseEnv la posición del gabinete
4. El owner utiliza AStar para calcular una ruta hacia el gabinete
5. HouseEnv procesa las acciones de movimiento y actualiza la posición del owner
6. El owner solicita abrir el gabinete
7. HouseView visualiza la apertura del gabinete
8. El owner toma el medicamento, reduciendo el inventario en HouseModel
9. HouseView visualiza la toma del medicamento
10. El owner envía un mensaje al robot notificando la medicación tomada
11. El robot actualiza sus creencias sobre medicamentos consumidos

Flujo de modificación de pautas

1. El owner decide modificar una pauta de medicación (por ejemplo, durante la noche)
2. El owner actualiza sus creencias con la nueva pauta
3. El owner envía un mensaje al robot con la pauta actualizada
4. El robot recibe el mensaje y actualiza sus creencias
5. El robot reorganiza su planificación de entregas según las nuevas pautas

Gestión de datos e información

El sistema domótico para la gestión de medicamentos requiere manejar diversos tipos de datos e información para su correcto funcionamiento. A continuación, se describe cómo se gestionan estos datos en el sistema.

Modelo de datos para el inventario de medicamentos

El inventario de medicamentos se implementa mediante una estructura de datos que asocia nombres de medicamentos con sus cantidades disponibles. La estructura básica es la siguiente:

Inventario inicial de medicamentos

Cada medicamento tiene asociados los siguientes atributos:

- Nombre: Identificador único del medicamento
- Cantidad: Número de unidades disponibles (todos empiezan con 20).
- Ubicación.

Gestión de pautas de medicación

Las pautas de medicación definen cuándo y con qué frecuencia se toma cada medicamento. Básicamente son:

Pautas de medicación iniciales

Medicamento	Hora	Frecuencia
Paracetamol	10:00	Cada 6 horas
Ibuprofeno	12:00	Cada 6 horas
Lorazepam	22:00	Cada 23 horas
Aspirina	17:00	Cada 8 horas
Fent	15:00	Cada 2 horas

Cada pauta tiene asociados los siguientes atributos:

- Medicamento: Nombre del medicamento a administrar
- Hora: Hora inicial de administración
- Frecuencia: Intervalo entre administraciones (en horas)

Las pautas se implementan como hechos en el conocimiento del agente owner.

El owner transmite estas pautas al robot al inicio de la simulación, y el robot las almacena como creencias propias. Cuando el owner modifica una pauta, envía la actualización al robot para mantener la coherencia.

Persistencia de información

El sistema no implementa persistencia permanente de datos, ya que la información se mantiene en memoria durante la ejecución de la simulación. Sin embargo, se implementa una persistencia lógica dentro de la simulación, mediante:

1. **Creencias de los agentes**: Los agentes mantienen creencias sobre el estado del mundo, incluyendo inventario, pautas y medicaciones tomadas
2. **Estado del modelo**: HouseModel mantiene el estado actual del entorno, incluyendo posiciones de objetos y cantidades de medicamentos
3. **Registro de actividad**: El sistema mantiene un registro temporal de las acciones realizadas durante la simulación

Representación interna del entorno

El entorno doméstico se representa internamente mediante:

1. **Matriz de ocupación**: Matriz bidimensional que indica qué celdas están ocupadas por objetos o agentes
2. **Mapa de habitaciones**: Estructura que define las diferentes habitaciones y sus límites
3. **Catálogo de muebles**: Colección de objetos que representan los muebles, con sus posiciones y dimensiones
4. **Posiciones de agentes**: Coordenadas actuales de cada agente en el entorno
5. **Estado de objetos interactivos**: Información sobre el estado de objetos como gabinetes (abiertos/cerrados)

Esta representación permite:

- Determinar si un movimiento es válido (no colisiona con obstáculos)
- Localizar objetos específicos como el gabinete de medicamentos
- Identificar la habitación en la que se encuentra un agente
- Visualizar correctamente el entorno y sus elementos

Conclusiones

En cuanto a las conclusiones, el proyecto ha dejado en evidencia que la capacidad de los agentes de percibir su entorno, tomar decisiones autónomas, y coordinar sus acciones es

especialmente útil en escenarios que requieren adaptabilidad y reactividad a circunstancias en constante cambio. La arquitectura BDI implementada con Jason ha dado un marco muy robusto para diseñar agentes con comportamiento complejo, permitiendo separar de forma clara y modular el conocimiento, los objetivos y los planes de acción.

Gestión de interacciones entre agentes y entorno

El desarrollo ha evidenciado la complejidad de gestionar las interacciones entre agentes y entorno, especialmente en aspectos como:

- Sincronización de acciones cuando varios agentes interactúan con el mismo objeto
- Representación y actualización coherente del estado de objetos manipulables
- Visualización clara de interacciones para facilitar la comprensión del usuario
- Gestión de colisiones y conflictos en el acceso a recursos compartidos

La solución implementada mediante un entorno centralizado que coordina estas interacciones ha resultado efectiva.