CEA Comité Español de Automática
XLV JORNADAS AUTOMÁTICA MÁLAGA 2024
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Control de un robot social mediante modelos de comportamiento para el cuidado de personas mayores

Merino-Fidalgo, S.[a,*], Sánchez-Girón, C.[a], Zalama, E.[a,b], Gómez-García-Bermejo, J.[a,b], Duque-Domingo, J.[a]

[a] *ITAP-DISA, Universidad de Valladolid, C/ Doctor Mergelina, 47011, Valladolid, España.*
[b] *CARTIF, Parque Tecnológico de Boecillo, 47151 Boecillo, Valladolid, España.*

To cite this article: Merino-Fidalgo, S., Sánchez-Girón, C., Zalama, E., Gómez-García-Bermejo, J., Duque-Domingo, J. 2024. Control of a social robot using behavioral models for elderly people care. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10787

Resumen

En los últimos años se ha experimentado un aumento de personas mayores que viven solas en sus casas o en situación de dependencia, que requieren un cuidado especial. En muchas ocasiones, ofrecer este servicio resulta complicado, puesto que un cuidador puede tener a su cargo a varias personas de diferentes localidades. Para solucionar este problema, el proyecto EIAROB trabaja para crear un ecosistema de inteligencia ambiental para el apoyo a los cuidados de larga duración en el hogar mediante uso de robots sociales. En este artículo se presenta una arquitectura encargada del control de un robot social mediante máquinas de estados finitos jerárquicas y árboles de comportamiento, dentro de una vivienda de una persona mayor para que dicho robot interactúe con ella, ejecutando planes mediante un ejecutor de acciones. Se exponen dos ejemplos de planes: dar los buenos días y recordar la medicación, realizados con ambos métodos y finalmente se presentan y comparan sus resultados.

Palabras clave: Sistemas de control embebidos y aplicaciones, Internet de las Cosas, redes de robots y sensores inteligentes, robots móviles, aspectos cognitivos de los sistemas de automatización y los seres humanos.

Control of a social robot using behavioral models for elderly people care

Abstract

In recent years there has been an increase in the number of elderly people living alone in their homes or in a situation of dependency, who require special care. In many cases, offering this service is complicated, since one caregiver may be in charge of several people from different locations. To solve this problem, the EIAROB project works to create an ambient intelligence ecosystem to support long-term care at home using social robots. This paper presents an architecture for the control of a social robot using hierarchical finite state machines and behaviour trees, inside an elderly person's home so that the robot interacts with the elderly person, executing plans by means of an action executor. Two examples of plans are presented: saying good morning and remembering medication, carried out with both methods, and finally their results are presented and compared.

Keywords: Embedded computer control systems and applications, Internet of Things, networks of robots and intelligent sensors, mobile robots, cognitive aspects of automation systems and humans.

1. Introducción

Controlar un robot constituye uno de los desafíos más complejos de la ingeniería moderna. Desde el comienzo de la robótica, los primeros autómatas estaban diseñados para realizar una única tarea, simplificando así su diseño y programación. Sin embargo, la necesidad de llevar a cabo acciones de mayor complejidad y la capacidad de crear robots multitarea más completos, precisó una mejora en la arquitectura de control y la introducción de nuevas técnicas que permitiesen manejar estas dificultades añadidas.

*Autor para correspondencia: sergio.merino.fidalgo@uva.es

Una de las primeras aproximaciones fue la máquina de estados finitos (FSM), un concepto ampliamente conocido que consiste en una serie de estados y las correspondientes transiciones (Figura 1), las cuales se caracterizan por una condición que lleva a cabo la transición y una acción que se realiza al ejecutar la transición (Ben-Ari et al., 2018).

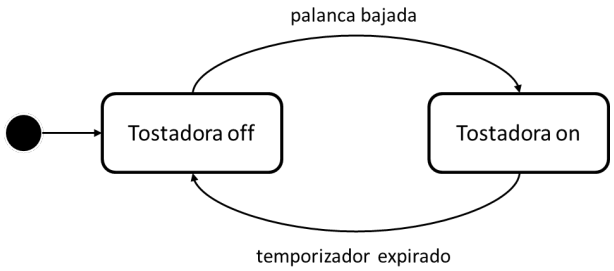

Figura 1: Diagrama de estados de una tostadora.

Las máquinas de estados finitos han sido la base de la programación de robots durante muchos años, debido a su buen comportamiento en entornos y acciones rígidas y predecibles. Sin embargo, este método presenta varios inconvenientes, como poca flexibilidad, escalabilidad compleja y respuesta incierta ante comportamientos impredecibles. Para solventar algunos de estos problemas surgieron las máquinas de estados finitos jerárquicas (HFSM), las cuales permiten que un estado pueda contener subestados, de tal forma que se ejecuta una máquina de estados dentro de otra máquina de estados (Girault et al., 1999). Esta nueva arquitectura mejora la reutilización del código pero sigue presente el problema de la adición o supresión de estados.

Los árboles de comportamiento (BTs) comenzaron a usarse en la industria de los videojuegos como una mejora de las máquinas de estados finitos en el control de Non-Player Characters (NPCs) (Mateas and Stern, 2002), (Flórez-Puga et al., 2008), y se definen como un método de estructurar el cambio entre distintas acciones mediante un esquema de árbol invertido (Figura 2) que contiene nodos lógicos y nodos de comportamiento (Colledanchise and Ögren, 2017), (Robertson and Watson, 2015).

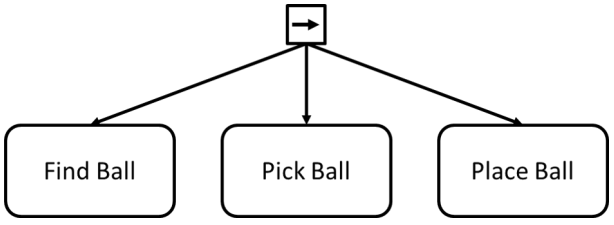

Figura 2: Diagrama de un árbol de comportamiento de una acción *Pick and Place*.

El árbol comienza con el nodo raíz generando *ticks* con una frecuencia determinada que permite la ejecución de los hijos de izquierda a derecha, los cuales pueden devolver *SUCCESS, FAILURE* o *RUNNING*, que indica si el resultado del nodo ha sido exitoso, fallido o sigue en ejecución. A continuación se muestra en la Figura 3 los tipos de nodos que componen un árbol de comportamiento y las condiciones que deben darse para devolver cada uno de los resultados posibles.

Node type	Symbol	Succeeds	Fails	Running
Fallback	?	If one child succeeds	If all children fail	If one child returns Running
Sequence	→	If all children succeed	If one child fails	If one child returns Running
Parallel	⇉	If $\geq M$ children succeed	If $> N - M$ children fail	else
Action	text	Upon completion	If impossible to complete	During completion
Condition	text	If true	If false	Never
Decorator	◊	Custom	Custom	Custom

Figura 3: Tabla de nodos de un árbol de comportamiento.

Las principales ventajas de los árboles de comportamiento frente a las máquinas de estados finitos son la modularidad, que permite añadir, quitar o reutilizar nodos con facilidad, un aumento de la reactividad al reaccionar de forma más rápida y eficiente a cambios y la facilidad de mantenimiento (Colledanchise and Ögren, 2017).

En los últimos años los árboles de comportamiento se han empezado a utilizar también en robótica social debido a la reactividad que ofrecen los robots sociales al interactuar con personas. Uno de los primeros proyectos en emplear esta arquitectura fue JIBO (Hodson, 2014), un pequeño robot social que puede reconocer caras y voces, jugar y compartir información gracias a una arquitectura basada en árboles de comportamiento. Cooper and Lemaignan (2022) emplearon el robot ARI (Cooper et al., 2020) para trabajar con personas mayores empleando dos árboles de comportamiento: uno cuando alguien se acercaba a interactuar con él, iniciaba una conversación y proponía juegos y distintas actividades, mientras que el segundo árbol permitía desplazarse hasta la localización de la persona para emitir un recordatorio.

El presente artículo se centra en el control mediante distintos métodos de un robot social dentro de la vivienda de una persona mayor para que interactúe con él (Sección 2) realizando planes mediante un ejecutor de acciones, los cuales son llevados a cabo por un robot social (Sección 3). Se exponen dos ejemplos de planes: dar los buenos días y recordar la medicación, realizados ambos empleando máquinas de estados finitos jerárquicos y árboles de comportamiento (Sección 4), y se presentan los resultados de estos, la comparativa entre ambos métodos y las líneas futuras en la Sección 5. Finalmente, las conclusiones se resumen en la Sección 6.

2. Marco de la investigación

En los últimos años, ha habido un notorio aumento en el número de personas mayores que viven solas en sus hogares, una tendencia que se espera que continúe creciendo en España, especialmente en la región de Castilla y León. Datos recientes señalan que el 26,48 % de la población en esta comunidad autónoma tiene más de 65 años (Instituto Nacional de Estadística, 2023), y aproximadamente uno de cada cuatro de ellos vive solo, lo que equivale a alrededor de 161.500 personas (Junta de Castilla y León, 2023). Además, la falta de contacto con otras personas puede generar un profundo sentimiento de soledad, lo que podría resultar en graves problemas para la salud, especialmente en personas de edad avanzada (Hoppmann et al., 2021).

La investigación presentada en este artículo forma parte del proyecto EIAROB (Ecosistema de Inteligencia Ambiental para el apoyo a los cuidados de larga duración en el hogar mediante el uso de robots sociales), cuyo objetivo princi-

pal es promover la autonomía y mejorar la calidad de vida de las personas mayores y dependientes. Este proyecto se enfoca en tres áreas principales: el desarrollo de un sistema de inteligencia ambiental para respaldar la vida independiente, que supervisará y monitoreará las actividades diarias; la creación de soluciones mecatrónicas y robóticas para la atención a las personas mayores; y el establecimiento de un observatorio experimental de robótica para la vida independiente en Castilla y León (EIAROB, 2023). El proyecto aborda diversos campos de investigación, desde aspectos innovadores como la robótica social y el uso de inteligencia artificial para la detección de personas y el reconocimiento de actividades, hasta cuestiones importantes como el respeto a la privacidad de las personas mayores mediante el uso de tecnología domótica no invasiva y la gestión de situaciones de emergencia.

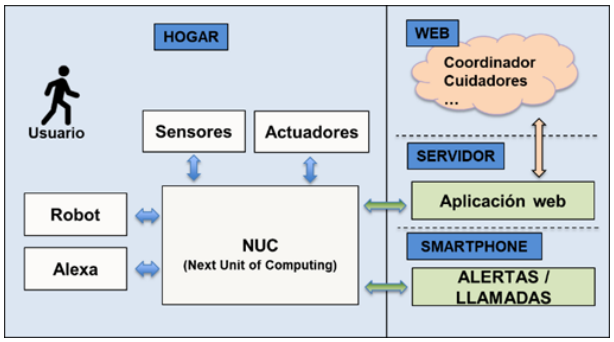

Figura 4: Esquema general del proyecto Eiarob.

En la Figura 4, se presenta un esquema simplificado del proyecto, que muestra cada una de las partes que lo conforman. La instalación del interior de la vivienda consta de un sistema domótico compuesto por un conjunto de sensores de reducido tamaño distribuidos por la casa para recopilar información sobre la vivienda y las condiciones y actividades del residente. Estos datos son recibidos por un pequeño ordenador NUC (Next Unit of Computing) instalado en la vivienda, el cual administra los actuadores, emplea un asistente de voz (Alexa) para comunicarse con el residente y controla el funcionamiento del robot social. El NUC, a su vez, se encarga de enviar notificaciones y realizar llamadas al móvil del cuidador responsable del residente, y se comunica con la aplicación web de un servidor. En este servidor se almacena toda la información de la vivienda, a la que acceden el coordinador y los cuidadores a través de una página web que permite visualizar información en tiempo real de la vivienda y el anciano, y programar actividades y acciones para que sean llevadas a cabo por el residente y el robot social, respectivamente (Merino-Fidalgo et al., 2023).

3. Ejecución de acciones

La detección de situaciones peligrosas y la programación de actividades por parte del cuidador, como por ejemplo, recordarle al residente que tome sus medicinas a una hora específica, requieren la ejecución de una contramedida que genera un mensaje MQTT, protocolo basado en un modelo publicador-suscriptor para el envío de mensajes (Mishra and Kertesz, 2020). Este mensaje está formado por el tema (topic) y la carga útil (payload) de la acción correspondiente. Esto se hace para corregir el estado de emergencia o para asegurar que el residente realice la actividad propuesta por el cuidador, como es el caso de recordarle que tome las medicinas si aún no lo ha hecho. La ejecución de estas acciones se gestiona desde un ejecutor de acciones que se ejecuta en el NUC. En la Figura 5 se presenta el proceso seguido por el ejecutor de acciones para la ejecución de las mismas. Cuando recibe un mensaje MQTT, el ejecutor de acciones analiza su contenido y genera un plan para llevar a cabo la acción requerida. Este plan se agrega a una cola de ejecución y se reordena en función de la prioridad asignada a los planes, que se determina según el tipo de acción y el usuario que la solicita (cuidador, administrador, residente, etc.). Si la cola de ejecución está vacía o la acción tiene la máxima prioridad y hay un plan de menor prioridad en ejecución (lo que conlleva el aborto del plan que se está llevando a cabo), el ejecutor de acciones inicia el plan y se encarga de su ejecución. Cuando este finaliza, el ejecutor verifica su estado final y envía un mensaje que indica si la acción se ha llevado a cabo correctamente o no. Luego, guarda esta información en la base de datos del servidor. Después de completar este proceso, el plan se elimina de la cola de ejecución. Si hay planes restantes en la cola, el ejecutor inicia el siguiente plan. En caso contrario, el ejecutor entra en un estado de reposo hasta que reciba un nuevo plan, si la cola de ejecución se encuentra vacía.

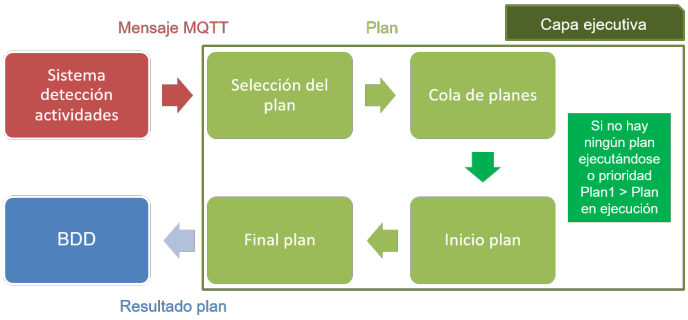

Figura 5: Esquema del proceso de ejecución de acciones.

Estos planes pueden ser acciones simples (apagar una luz o encender la calefacción) o acciones más complejas que requieren un control de su ejecución para adaptarse a las posibles variantes y cambios que pueden acontecer. El elemento que lleva a cabo estas acciones es el robot Temi, un robot social móvil (Temi V3, 2024) en el que se ha desarrollado una aplicación que ofrece una variedad de servicios, incluyendo ejercicios mentales y físicos, juegos de memoria y acertijos, videoconferencias con cuidadores o familiares e incluso una integración con ChatGPT (Kohnke et al., 2023) que proporciona información sobre cualquier tema que el usuario solicite. El control del robot también se puede llevar a cabo mediante comandos enviados a través de mensajes MQTT utilizando el bróker Mosquitto (Koziolek et al., 2020), instalado en el NUC de la vivienda. Este método es empleado por el ejecutor de acciones para dirigir al robot a una ubicación específica o para que reproduzca un mensaje.

4. Métodos propuestos

En la introducción de este artículo se han comentado los métodos empleados para llevar a cabo las acciones y planes propuestos, con el fin de comparar su dificultad a la hora de programar y de adaptarlos para ser compatibles con el ejecutor de acciones y, finalmente, su desempeño al ser ejecutados.

Las máquinas de estados finitos son un método ampliamente estudiado y desarrollado, por lo que su programación (empleando la librería Python-Statemachine (Macedo, 2023)) resulta sencilla inicialmente, puesto que, una vez definidos los estados y las transiciones, solo hay que indicar qué estado o estados van a continuación y las correspondientes transiciones. A continuación, se adapta para que se pueda ejecutar desde el ejecutor de acciones, tarea relativamente sencilla puesto que la máquina de estados se conforma de una única clase Python, por lo que se define una función de inicio y otra de finalización para que el ejecutor de acciones las llame. De este modo, cuando se recibe un mensaje MQTT para ejecutar una acción, se llama al plan asociado a la máquina de estados previamente programada, que posteriormente se inicia (creando una nueva conexión MQTT para que la máquina reciba y envíe mensajes con el robot) cuando comienza la ejecución del plan. Una vez finalizada la máquina de estados, el ejecutor de acciones comprueba si esta ha alcanzado un estado final, llama la función de la máquina para cerrar su conexión MQTT y elimina el plan. Para los planes se han utilizado máquinas de estados finitos jerárquicas, por lo que se han creado módulos que realizan una acción simple pero muy utilizada para poder ejecutarla en todos los planes que sea necesario.

Los árboles de comportamiento, por el contrario, son una arquitectura más novedosa en el mundo de la robótica social, y se empleó la librería Py_Trees (Stonier, 2023), en la que cada nodo constituye una clase Python formada generalmente por una función de inicio y una función *update*. A continuación, se crea una nueva clase con la distribución jerárquica de nodos para conformar el árbol a partir de los nodos selectores y secuenciales. Estos elementos junto con los decoradores, que proporcionan modificaciones comunes a los nodos subyacentes como, por ejemplo, invirtiendo el resultado, son utilizados para controlar la lógica del sistema y, por tanto, cómo debe ser su ejecución. Por último, para que el árbol de comportamiento también pueda ser llamado desde el ejecutor de acciones, se crea una clase en la que se realiza la conexión MQTT y se define el árbol de comportamiento al iniciar el plan, y se desconecta del bróker cuando este ha finalizado, de la misma forma que se realiza para las máquinas de estados. De igual forma que sucede con las máquinas de estados jerárquicas, los árboles de comportamiento pueden ejecutar otros árboles dentro de los nodos e incluso máquinas de estados finitos.

En el presente trabajo, el ejecutor de acciones se ha adaptado para poder llevar a cabo planes realizados tanto por máquinas de estados como por árboles de comportamiento de forma indistinta, ya que se ha adaptado a las particularidades que ambos métodos presentan. La mayor diferencia reside a la hora de controlar la ejecución del plan, donde en el caso de las máquinas de estados se ha definido una acción genérica que comprueba todos los estados y sus posibles transiciones y permite cambiar de estado si se cumplen las condiciones correspondientes. En el caso de los árboles de comportamiento, el ejecutor de acciones ejecuta un *tick*, el cual evalúa el nodo actual para actualizar su estado, *RUNNING* si sigue ejecutándose y *SUCCESS* o *FAILURE* cuando el nodo ha finalizado.

4.1. Ejemplos de acciones

A continuación se presentan dos ejemplos de planes que llevan a cabo acciones complejas, explicando el flujo de ejecución de cada uno teniendo en cuenta los contratiempos que pueden surgir durante la ejecución o las múltiples respuestas e interacciones que el usuario puede realizar. Ambos planes se han llevado a cabo mediante máquinas de estados y árboles de comportamiento, para comparar el desempeño de cada método en este tipo de situaciones.

4.1.1. Recordar medicinas

El plan de recordar medicinas lleva a cabo la acción de avisar al residente de que debe tomar la medicación correspondiente. Este plan se ejecuta si, existiendo una toma de medicinas programada por el cuidador, el algoritmo de reconocimiento de actividades no detecta que el residente la haya realizado.

La primera acción del plan consiste en enviar al robot a la ubicación más cercana a la posición del residente, información proporcionada por una métrica que devuelve en todo momento la posición de la persona dentro de la casa. Si el robot alcanza su destino, pregunta al residente si ha tomado sus medicinas y espera su respuesta. Si no responde, repite la pregunta y finaliza el plan si sigue sin obtener respuesta. Si el residente dice que sí, el robot expresa unas palabras de agradecimiento y termina la acción. Si no ha tomado la medicación, el robot le recuerda la medicación específica, repitiendo el recordatorio hasta que el residente la tome o se supere el tiempo máximo, finalizando entonces la tarea. Cuando se completa una acción, se reproduce un recordatorio o mensaje de interés para el residente, como indicar cómo jugar a un juego.

En la Figura 7 se muestra un diagrama de estados de la ejecución del plan, con cada uno de sus estados y transiciones. Para mandar el robot a una ubicación se utiliza un superestado que ejecuta una máquina de estados que controla el desplazamiento del dispositivo, teniendo en cuenta situaciones como que no pueda llegar al destino o que el residente interactúe con él. La Figura 6 ilustra el esquema del árbol de comportamiento para la misma acción, donde el superestado para gestionar el movimiento del robot se ejecuta en los nodos correspondientes a la rama izquierda del diagrama.

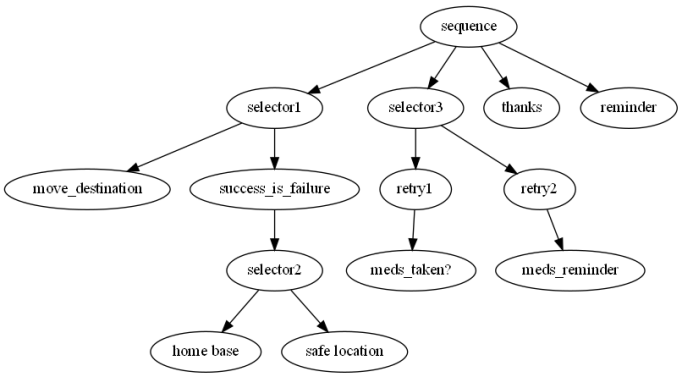

Figura 6: Diagrama del árbol de comportamiento de la acción *Recordar medicinas*.

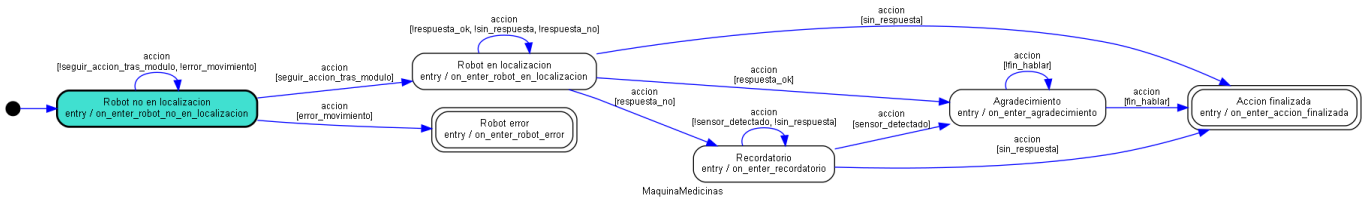

Figura 7: Diagrama de la máquina de estados de la acción *Recordar medicinas*.

4.1.2. Buenos días

El plan de buenos días es un plan recursivo que se ejecuta diariamente a una hora determinada por la mañana para verificar que el residente está bien y proporcionarle información de interés. Se activa cuando se alcanza la hora estipulada por el cuidador y se detecta que el residente sigue durmiendo en su cama.

El plan comienza enviando al robot a la habitación del usuario y, en el caso de que llegue, el robot pregunta si ha dormido bien. Si la respuesta es afirmativa, ofrece escuchar noticias, el clima o un dato interesante. Si el usuario dice que no ha dormido bien, el robot pregunta si quiere llamar a su contacto de emergencia, realizando la llamada si el usuario acepta y finalizando el plan si no. Si el usuario no responde después de dos intentos, el robot activa la cámara para buscar al residente usando un algoritmo de visión basado en Mediapipe (Singh et al., 2022). Si no encuentra a nadie, regresa a su base y finaliza el plan, mientras que en el caso de que encuentre al residente, pregunta si se encuentra bien. En caso afirmativo, tras detectar una persona en la foto recibida como se muestra en la Figura 8, le propone alguna actividad de las anteriormente mencionadas; si dice que no o no hay respuesta, ofrece hacer una videollamada al contacto de emergencia.

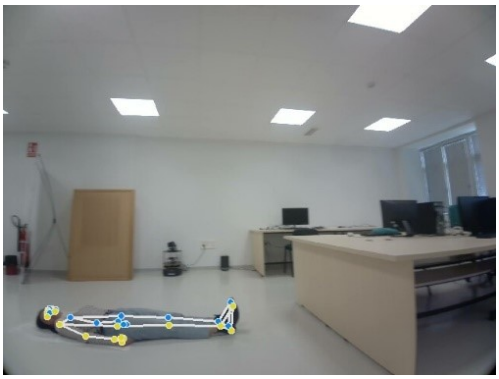

Figura 8: Imagen en la que se ha detectado una persona.

5. Resultados

Para comparar el desempeño de los dos métodos empleados se ha acondicionado el espacio de trabajo para asemejarse a una vivienda típica de una persona mayor, para así poder reproducir los planes de la forma más realista posible. Las diferencias entre máquinas de estados finitos y árboles de comportamiento se han testado mediante múltiples pruebas de los planes anteriormente mencionados, siguiendo un procedimiento para llevar a cabo cada una de las posibilidades y variaciones que los planes ofrecen, así como la introducción de factores extraños, obstáculos o situaciones no contempladas directamente durante la programación, para un total de 130 pruebas para las máquinas de estados y otras 130 para los árboles de comportamiento.

Tabla 1: Resultados de las máquinas de estados finitos

	Bien	*Mal*	*Total*	*Porcentaje de éxito*
Medicinas	49	1	50	98 %
Buenos días	77	3	80	96.25 %
Total	126	4	130	96.92 %

Tabla 2: Resultados de los árboles de comportamiento

	Bien	*Mal*	*Total*	*Porcentaje de éxito*
Medicinas	47	3	50	94 %
Buenos días	74	6	80	92.5 %
Total	121	9	130	93.07 %

La tabla 1 muestra los resultados obtenidos de las máquinas de estados finitos, obteniendo un porcentaje de éxito conjunto muy positivo, cerca del 97 %. Por su parte, la tabla 2 hace referencia a las pruebas realizadas con los árboles de comportamiento, que arroja unos resultados similares aunque ligeramente más bajos, en torno al 93 %.

En cuanto a cada uno de los planes, el de recordar medicinas fue aquel que mejor resultados obtuvo para ambos métodos, alcanzando el 98 % de porcentaje de éxito con la máquina de estados y un 94 % con el árbol de comportamiento, con solo dos casos de éxito menos. El plan de buenos días, debido a su mayor complejidad y a la amplia variedad de opciones y respuestas, obtuvo unos resultados peores, de un 96.25 % para la máquina de estados y un 92.5 % en el caso del árbol de comportamiento.

Al analizar los errores y fallos acontecidos, se observa que, de los 13 casos totales en los que la ejecución no fue exitosa, más de la mitad (siete) fueron por problemas al capturar la respuesta del usuario, principalmente porque no captó bien lo que se dijo o no entendió el sentido de la respuesta (por ejemplo, se contestó ”no me puedo quejar” a la pregunta de si ha dormido bien y el sistema lo interpretó como una respuesta negativa). En cuatro ocasiones se produjeron fallos de la conexión del robot con el bróker Mosquitto, ya que a veces se desconectaba tras estar mucho tiempo sin utilizarse, problema en el que ya se trabaja para lograr una conexión más robusta. En una ocasión, durante la búsqueda de persona tras no detectar respuesta, el sistema detectó como persona un abrigo, dando lugar a un falso positivo.

En términos de funcionamiento, tanto ambos modelos de comportamiento ofrecieron un desempeño similar, ofreciendo

gran robustez y seguridad salvo en una de las pruebas realizadas en el plan de buenos días hecho con árboles de comportamiento, donde un nodo devolvió un resultado indeseado tras introducir una situación no contemplada inicialmente. Por lo tanto, ambos métodos resultan perfectamente adecuados para el desempeño del control del robot, por lo que se ha decidido emplear máquinas de estados y árboles de comportamiento maximizando sus puntos fuertes, con el uso de las primeras para acciones más sencillas y rígidas, y los segundos para planes más complejos, con mayor variabilidad o que requieran una adaptación personalizada para cada usuario.

Las futuras líneas de investigación comprenden la mejora del funcionamiento del sistema y seguir perfeccionando los planes para lograr que sean más completos y robustos y que se puedan adaptar a cada usuario. Estas mejoras serán presumiblemente más sencillas de llevar a cabo en los árboles de comportamiento, debido a su facilidad para reutilizar código, así como añadir o eliminar nodos, tarea mucho más compleja de realizar en el caso de las máquinas de estados finitos por el número de posibles transiciones en cada uno de los estados. Finalmente, el objetivo es desplegar estos planes en viviendas reales y testarlos de forma intensiva.

6. Conclusiones

En este artículo se han presentado dos metodologías para gestionar un sistema de control de un robot social para ser desplegado en la vivienda de una persona mayor con el fin de mejorar su calidad de vida, así como permitir a su cuidador comunicarse con ella y programar planes que sean llevados a cabo por el robot. Para comparar ambas arquitecturas, se han realizado pruebas con dos acciones, recordar las medicinas y dar los buenos días, programadas con máquinas de estados finitos jerárquicas y árboles de comportamiento, cuyos resultados demostraron que no existen grandes diferencias en la ejecución y robustez entre los métodos propuestos. Tras analizar otros aspectos, se llegó a la conclusión de que las máquinas de estados finitos están ampliamente establecidas en la robótica como una metodología robusta y su programación es conocida y está completamente desarrollada, por lo que resultan la mejor opción para iniciarse en el control de robots y para tareas sencillas. Sin embargo, los árboles de comportamiento, que comenzaron a utilizarse en el control de robots recientemente, ofrecen mayores posibilidades debido a su facilidad de mantenimiento y de modificar o ampliar el árbol, gracias a la modularidad intrínseca que presentan, a pesar de que, inicialmente, su programación puede resultar ligeramente más compleja, puesto que es necesario crear el árbol jerárquico con la lógica de ejecución antes de programar. Como líneas futuras se busca la mejora continua de los planes para que sean más robustos y personalizables con el fin de implantarlos y testarlos en viviendas reales.

Agradecimientos

La investigación que se presenta en este trabajo ha recibido financiación del proyecto ROSOGAR PID2021-123020OB-I00 financiado por MCIN/ AEI / 10.13039/501100011033 / FEDER, UE, y del proyecto EIAROB Financiado por Consejería de Familia de la Junta de Castilla y León - Next Generation EU.

Referencias

Ben-Ari, M., Mondada, F., Ben-Ari, M., Mondada, F., 2018. Finite state machines. Elements of Robotics, 55–61.
DOI: 10.1007/978-3-319-62533-1_4

Colledanchise, M., Ögren, P., 8 2017. Behavior trees in robotics and ai: An introduction. Behavior Trees in Robotics and AI.
DOI: 10.1201/9780429489105

Cooper, S., Di Fava, A., Vivas, C., Marchionni, L., Ferro, F., 2020. Ari: the social assistive robot and companion. In: 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). pp. 745–751.
DOI: 10.1109/RO-MAN47096.2020.9223470

Cooper, S., Lemaignan, S., 2022. Towards using behaviour trees for long-term social robot behaviour. In: 2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI). pp. 737–741.
DOI: 10.1109/HRI53351.2022.9889662

EIAROB, 2023. Eiarob. https://www.cartif.es/eiarob/.

Flórez-Puga, G., Gómez-Martín, M., Díaz-Agudo, B., González-Calero, P. A., 2008. Dynamic expansion of behaviour trees. In: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment. Vol. 4. pp. 36–41.
DOI: 10.1609/AIIDE.V4I1.18669

Girault, A., Lee, B., Lee, E., 1999. Hierarchical finite state machines with multiple concurrency models. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18 (6), 742–760.
DOI: 10.1109/43.766725

Hodson, H., 2014. The first family robot. New Scientist 223 (2978), 21.
DOI: https://doi.org/10.1016/S0262-4079(14)61389-0

Hoppmann, C. A., Lay, J. C., Pauly, T., Zambrano, E., 4 2021. Social isolation, loneliness, and solitude in older adulthood. The Handbook of Solitude, 178–189.
DOI: 10.1002/9781119576457.CH13

Instituto Nacional de Estadística, 2023. Proporción de personas mayores de cierta edad por comunidad autónoma(1451). https://www.ine.es/jaxiT3/Datos.htm?t=1451#!tabs-tabla.

Junta de Castilla y León, 2023. La junta activa un teléfono gratuito para detectar, atender y combatir el aislamiento social de los mayores. https://comunicacion.jcyl.es/web/jcyl/Comunicacion/es/Plantilla100Detalle/1284721258421/NotaPrensa/1285343829939/Comunicacion.

Kohnke, L., Moorhouse, B. L., Zou, D., 2023. Chatgpt for language teaching and learning. RELC Journal 54 (2), 537–550.
DOI: 10.1177/00336882231162868

Koziolek, H., Grüner, S., Rückert, J., 2020. A comparison of mqtt brokers for distributed iot edge computing. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12292 LNCS, 352–368.
DOI: 10.1007/978-3-030-58923-3_23

Macedo, F., 2023. Python state machine — python state machine 2.2.0 documentation.
URL: https://python-statemachine.readthedocs.io/en/latest/index.html

Mateas, M., Stern, A., 2002. A behavior language for story-based believable agents. IEEE Intelligent Systems 17 (4), 39–47.
DOI: 10.1109/MIS.2002.1024751

Merino-Fidalgo, S., Zalama, E., Gómez-García-Bermejo, J.and Duque-Domingo, J., Gómez, R., Viñas, P., García, D., Urueña, H., 7 2023. Sistema de monitorización no intrusiva para vivienda de personas mayores. Jornadas Nacionales de Robótica y Bioingeniería 2023: Libro de actas, 115–121.
DOI: 10.20868/UPM.BOOK.74896

Mishra, B., Kertesz, A., 2020. The use of mqtt in m2m and iot systems: A survey. IEEE Access 8, 201071–201086.
DOI: 10.1109/ACCESS.2020.3035849

Robertson, G., Watson, I., 2015. Building behavior trees from observations in real-time strategy games. In: 2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA). pp. 1–7.
DOI: 10.1109/INISTA.2015.7276774

Singh, A. K., Kumbhare, V. A., Arthi, K., 2022. Real-time human pose detection and recognition using mediapipe, 145–154.
DOI: 10.1007/978-981-16-7088-6_12

Stonier, D., 2023. Py trees — py_trees 2.2.3 documentation.
URL: https://py-trees.readthedocs.io/en/devel/index.html

Temi V3, 2024. Temi v3. https://www.robotemi.com/product/temi/.