PLANTILLA PARA LA PROGRAMACIÓN EN C# DE AGENTES SOFTWARE MÓVILES

Article

4 authors, including:

Juan-Luis Posadas-Yagüe
Polytechnic University of Valencia
92 PUBLICATIONS **491** CITATIONS

SEE PROFILE

Jose-Luis Poza-Luján
Polytechnic University of Valencia
120 PUBLICATIONS **517** CITATIONS

SEE PROFILE

Jose Enrique Simo
Polytechnic University of Valencia
156 PUBLICATIONS **1,141** CITATIONS

SEE PROFILE

PLANTILLA PARA LA PROGRAMACIÓN EN C# DE AGENTES SOFTWARE MÓVILES

Sixto Franco Martínez
DISCA, UPV, Camino de Vera s/n. 46022 Valencia, Spain, sixframa@eui.upv.es

Juan-Luis Posadas Yagüe
DISCA, UPV, Camino de Vera s/n. 46022 Valencia, Spain, jposadas@disca.upv.es)

José-Luis Poza Luján
DISCA, UPV, Camino de Vera s/n. 46022 Valencia, Spain, jopolu@disca.upv.es)

José Simó Ten
DISCA, UPV, Camino de Vera s/n. 46022 Valencia, Spain, jsimo@disca.upv.es)

Resumen

En este artículo se presenta una plantilla para la programación de agentes software móviles basados en la arquitectura SC-Agent. Dicha arquitectura se sustenta sobre un sistema de comunicaciones que permite la interacción entre los agentes mediante la escritura/lectura de objetos compartidos en una estructura de pizarra distribuida. La plantilla desarrollada proporciona al agente su estructura de datos y de hilos de ejecución, así como el código necesario para las comunicaciones, permitiendo al programador centrarse únicamente en aquellos aspectos que caracterizan a su agente. La plantilla se ha implementado en C# y se ha validado mediante el desarrollo de diferentes agentes.

Palabras Clave: Agentes *Software* Móviles, Sistemas Distribuidos, Sistemas de Tiempo Real.

1 INTRODUCCIÓN

De la misma manera que existen plantillas para la creación de nuevas clases, interfaces y otras estructuras *software*, también cabe la posibilidad de proporcionar nuevas plantillas para el desarrollo de aquellos componentes *software* que surgen con las nuevas tecnologías. En este artículo se presenta una plantilla que sirve como base para la creación de cualquier agente *software* móvil [1,6], independientemente de la función final para la que se destinará dicho agente. Esta plantilla está basada en el sistema de comunicaciones que proporciona la arquitectura SC-Agent [5], tal como se describe en la sección siguiente, y se ha implementado mediante el lenguaje de programación C#.

Independientemente de la función para la que se diseña un agente móvil, éste posee unas características que se pueden considerar comunes a cualquiera de ellos. El principal objetivo de este trabajo ha sido obtener todas las características comunes y que se repiten en todos los agentes para producir una plantilla que sirva de base a cualquier programador que desee crear nuevos agentes. La función de la plantilla obtenida consiste en liberar al programador del desarrollo de los aspectos básicos del agente, tal como su estructura de hilos de ejecución o el código necesario para las comunicaciones con otros agentes o para su movilidad, permitiéndole centrarse únicamente en el desarrollo de la tarea o función que llevará a cabo su agente.

El artículo está estructurado en cinco apartados, a continuación se describe de forma general el sistema de comunicaciones utilizado para el desarrollo de los agentes, seguidamente se describe la estructura de la plantilla desarrollada, a continuación las pruebas realizadas junto a los resultados obtenidos y, finalmente, las conclusiones.

2 SISTEMA DE COMUNICACIONES

El sistema de comunicaciones utilizado para la comunicación entre los agentes recibe el nombre de sistema RT-SCore [4] y está basado en el uso de una pizarra [2] de objetos distribuida que permite la interoperatividad entre los agentes independientemente de su ubicación física y lenguaje de implementación. Los agentes se comunican escribiendo y leyendo sobre los objetos o variables de la pizarra empleando una interfaz común denominada FSA [3]. Cada objeto o variable de la pizarra distribuida puede verse como un canal lógico

de transmisión de información (por ejemplo, el objeto o variable correspondiente al valor de un sensor de infrarrojos o el objeto o variable correspondiente al valor de la velocidad aplicada a un motor).

Existen dos formas de interacción o comunicación entre agentes a través de la pizarra. Ambas pueden coexistir requiriéndose la existencia por cada agente de al menos dos variables u objetos asociados (figura 1):

- Un objeto o variable para la información del agente. La información que el agente genera la escribe en su objeto asociado en la pizarra y quien desea acceder a dicha información sólo tiene que leer el objeto.

- Otro objeto o variable para enviar mensajes o datos al agente. Cuando otro agente desea enviar un mensaje o datos al agente, escribe el mensaje o datos en el objeto de la pizarra asociado al agente y éste recibe la información enviada mediante la lectura del objeto.

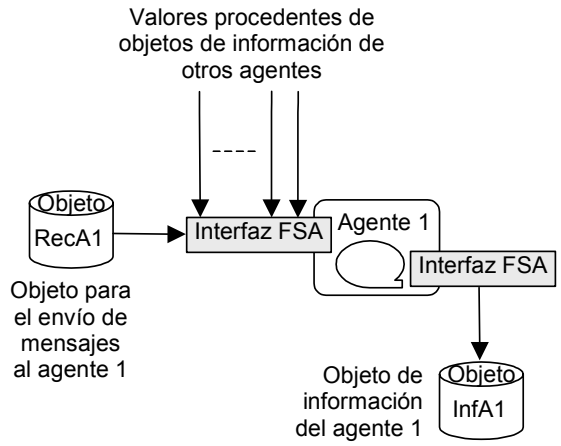

Figura 1: Sistema de Comunicaciones RT-SCore. Objetos de la pizarra distribuida asociados a cada agente.

Para las lecturas de los objetos el sistema ofrece un medio de notificación automática basado en eventos. Además, el sistema RT-SCore proporciona junto a los datos información temporal de los mismos, de forma que los agentes pueden validar la información que reciben en función de su antigüedad.

El sistema de comunicaciones RT-SCore requiere su ejecución en cada uno de los computadores del sistema. Igualmente, para permitir el movimiento de los agentes, se requiere la ejecución del agente REA (receptor y ejecutor de agentes) en cada uno de los nodos del sistema. Este agente se encarga de recibir el código de los agentes que se mueven y de ejecutarlos en su nueva localización. Un agente puede moverse con sólo escribir su código en el objeto o variable de la pizarra que está asociado al agente REA del nodo al que va a moverse.

En líneas generales, el sistema RT-SCore permite la recepción automática de los datos, los caracteriza temporalmente proporcionando su antigüedad y además ofrece una interfaz para la transmisión y localización de los agentes *software*.

3 PLANTILLA PARA UN AGENTE MÓVIL

El objetivo es obtener una plantilla general que pueda ser usada en la creación de nuevos agentes *software* móviles. Durante el desarrollo de esta plantilla se ha seguido un patrón general que se ajusta a la mayoría de los agentes móviles basados en el sistema RT-SCore.

La primera tarea para la creación de la plantilla consiste en analizar cuáles son todas las partes comunes a todos los agentes. Básicamente la misión desempeñada por un agente será conectarse a una o varias variables del sistema de comunicaciones (a las cuales otros agentes también podrán estar conectados), realizar una continua revisión del valor de las mismas, procesar periódicamente los nuevos valores que se obtengan, generar a partir de ellos los resultados propios del agente (en función de la tarea a realizar) y escribir dichos resultados en las variables del sistema asociadas al agente para que otros agentes puedan obtener la información generada.

Se han desarrollado dos versiones de la misma plantilla, diferenciadas por el número de hilos de ejecución que utilizan. Existe una versión monohilo y otra multihilo. En la versión mono-hilo el agente ejecuta de forma periódica un único hilo que se encarga del procesamiento de todas las variables del sistema de comunicaciones a las que el agente está conectado y de la generación de todos los resultados. En la versión multi-hilo el agente dispone de varios hilos con distintos periodos de ejecución que pueden repartirse el tratamiento de las distintas variables del sistema de comunicaciones. Con esta versión cada hilo puede generar una parte de los resultados del agente. Esta versión es adecuada cuando el agente implementa distintas funcionalidades o comportamientos donde los resultados de cada uno de ellos dependen de diferentes variables o deben obtenerse con distintos periodos.

El patrón de agente determinado por la plantilla es el siguiente:

- Conectarse a las variables del RT-Score

    - Obtener acceso al sistema de comunicaciones.
    - Vincular el método a ejecutar cuando el valor de cada variable cambie.
- Lanzar hilo o hilos de ejecución
    - Vincular con cada uno de los eventos o excepciones posibles, el código o método correspondiente del agente.
- Ejecución periódica los hilos

La plantilla también proporciona la implementación del método que permite al agente su movimiento a otro nodo a través del sistema de comunicaciones. El programador sólo tiene que realizar una llamada a dicho método para que el agente se mueva al nodo cuya dirección IP se especifica como parámetro en la llamada.

3.1 EXCEPCIONES

Como en todo programa que se desarrolla, pueden surgir situaciones que deben ser controladas por el programador. Los agentes no son una excepción y es por ello que durante su ejecución pueden aparecer situaciones o excepciones que deben resolverse.

En la plantilla se ha introducido el tratamiento de las siguientes excepciones:

- Excepcion_PerdidaDatos
- Excepcion_Antiguedad
- Excepcion_NoNuevosDatos
- Excepcion_Periodo

Cada una de las distintas excepciones trata un caso distinto y se ejecutarán en momentos y situaciones distintas. Las excepciones no se encuentran implementadas, ya que la tarea a realizar cuando se produce alguna excepción depende del tipo de excepción y del propósito del agente desarrollado, por eso el programador será el encargado de personalizar cada una de las excepciones en el caso de que el agente lo requiera.

Las excepciones codificadas son las relacionadas con la pérdida de datos, el periodo de ejecución, la antigüedad máxima de los datos y los datos para procesar. La excepción "Excepcion_PerdidaDatos" se lanza cada vez que entre un procesamiento y otro de los datos ha habido algún dato que no se ha procesado. La excepción "Excepcion_Antiguedad" se lanza cada vez que los datos que se están procesando han superado la máxima antigüedad permitida para esos datos. La excepción "Excepcion_NoNuevosDatos" se lanza cada vez que el hilo de ejecución entra para procesar los datos, pero sin embargo no hay nuevos datos a procesar, ya que los datos actuales ya han sido tratados y no se han recibido nuevos para ser tratados. La excepción "Excepcion_Periodo" se lanza cuando, una vez el hilo comprueba el tiempo de ejecución empleado para el procesamiento de los datos, este tiempo supera al periodo que el hilo tiene asignado.

Cabe recalcar que las excepciones en la plantilla se encuentran definidas para cada uno de los hilos. De esta forma, dependiendo del hilo (en la versión multihilo cada hilo puede implementar un comportamiento o tarea distinta) que lance la excepción ésta se podrá tratar de manera diferente. Así el programador puede personalizar el tratamiento de excepciones dependiendo del comportamiento que la genera (cada hilo se ocupa del tratamiento de un conjunto diferente de variables del sistema de comunicaciones) y no limitarle a tener que implementar un tratamiento estándar de excepciones que sirva para todas las variables con las que los distintos hilos de ejecución tratan.

3.2 HILOS DE EJECUCIÓN

La plantilla describe el esqueleto central del agente, estableciendo la conexión a las variables de RT-SCore que necesita, proporcionando las excepciones posibles y lanzando a ejecución los hilos. Estos últimos son los que realizan el trabajo central del agente. Una vez creado el agente y lanzados los hilos, el agente únicamente se dedica a esperar a que los hilos realicen su trabajo. Así pues, en los hilos se concentra la mayor parte del procesamiento y es aquí, dentro del hilo, donde el programador deberá introducir el código necesario para que el agente que desea implementar lleve a cabo correctamente su objetivo.

Por su parte, la estructura de los hilos que se lanzarán también seguirá un patrón común a todos ellos, diferenciándose unos de otros por su parte central o parte de procesamiento que diferirá una de otras dependiendo del objetivo a conseguir. La diferencia entre la versión monohilo y la multihilo consistirá en que empleando la versión monohilo sólo habrá que implementar la parte de procesamiento de un único hilo, en cambio si se emplea la multihilo se necesitará implementar tantas partes de procesamiento como hilos distintos se ejecuten.

El patrón que seguirá cada hilo de ejecución será el siguiente:

- Tomar tiempo (T1) antes de empezar el procesamiento
- Realizar procesamiento
    - Si no hay datos a procesar
        - Lanzar excepción "Excepcion_NoNuevosDatos"
    - Si hay datos a procesar

        - Acceso exclusivo a los datos procedentes de RT-SCore
        - Si perdida de datos
            - Lanzar excepción "Excepcion_PerdidaDatos"
        - Cambiar el valor de NuevosDatos a "false"
        - Cambiar el valor de PerdidaDatos a "0"
        - Si AntiguedadDatos es mayor que MAXIMA_ANTUGUEDAD
            - Lanzar excepción "Excepcion_Antiguedad"
        - *Ejecución del código específico del agente. Código proporcionado por el programador.*
        - Escribir los resultados obtenidos en el RT-SCore
- Tomar tiempo (T2) después de terminar el procesamiento
- Calcular la diferencia entre T1 y T2
    - Si la diferencia es mayor que el periodo
        - Lanzar excepción "Excepcion_Periodo"
- En el caso de no superar el periodo, dormir el hilo durante el tiempo de periodo no consumido en la ejecución, tiempo que se obtendrá de la diferencia: Periodo-(T2-T1).

El patrón mostrado anteriormente se repetirá mientras el agente no se cierre o se mueva a otro ordenador.

Para hacer uso de los hilos de ejecución hay que incorporar la librería de hilos proporcionada por .NET, para ello se usa de la directiva "using System.Threading". Con esto se pueden usar todos los métodos que .NET ofrece para la gestión de los hilos. La parte principal de los hilos consiste en un método que habrá que implementar y vincular al hilo. Este método será el método que el hilo ejecutará cuando se lance, recibe el nombre de "RunMethod". En este método se encuentra la implementación de los pasos que se deben seguir para que el agente consiga llevar a cabo el objetivo para el que ha sido creado. El método "RunMethod" seguirá la estructura que ha sido descrita anteriormente como patrón de comportamiento.

4 PRUEBAS REALIZADAS

Para validar la plantilla obtenida se expondrá el desarrollo de un agente basado en ella.

Uno de los usos que se da a los agentes es el control de robots. Se cuenta con un robot y la manera de controlar todas las funcionalidades de éste es dividir el sistema en partes más pequeñas (subsistemas) y menos complejos que son más fáciles y rápidos de desarrollar. Cada uno de estos subsistemas se trata de un agente, de modo que se desarrollan un conjunto de agentes que implementan cada una de las funcionalidades del robot, uno controla el movimiento, otro la velocidad y así todas las funcionalidades necesitadas por el robot. Una vez se cuenta con todos los agentes, de la interacción de todos ellos se obtiene el sistema global que controla la totalidad del robot. El objetivo del agente que se va a desarrollar consiste en implementar una funcionalidad del robot relacionada con la batería que suministra energía al robot. Dicho Agente se encargará de controlar el tiempo de batería que resta y, dependiendo de la posición donde se encuentra el robot y de la batería restante, deberá decidir si el robot debe regresar a donde se encuentra el recargador de la batería, para recargar la batería, y poder continuar con su correcto funcionamiento, o por el contrario si con la batería que resta todavía no existe la necesidad de volver a recargar la batería.

El agente deberá tomar la decisión de hacer regresar al robot o no dependiendo de la cantidad de batería restante y de la posición actual del robot, es decir, que dependerá de la información que aporten las variables que proporcionan la cantidad de batería restante y la posición del robot. Por ello, bastará con que el agente se conecte a dos variables del sistema de comunicaciones RT-SCore, la variable cuyo contenido es la cantidad de batería y la variable cuyo contenido es l aposición del robot. Aunque el agente a desarrollar se conecta a más de una variable, el cambio del valor de cualquiera de ellas implica la ejecución del mismo comportamiento (el cálculo de un resultado que indique mover o no el robot hacia el recargador de bateria).

Como únicamente hay que monitorizar dos variables implicadas en un único comportamiento para el cálculo de un resultado global, la versión de la plantilla más adecuada para el agente *software* planteado es la monohilo.

4.1 RESULTADOS

Los resultados obtenidos por el agente desarrollado confirman el correcto funcionamiento de la plantilla diseñada, demostrando que con el uso de la misma la creación de nuevos agentes es una tarea sencilla.

A continuación puede observarse de manera gráfica los resultados y funcionamiento del agente desarrollado para el control de la batería.

En la primera imagen (figura 2) puede observarse la situación en la que existe suficiente batería en el robot, de manera que éste sigue su camino sin interrupciones.

En la siguiente imagen (figura 3) puede observarse que el agente ha detectado que la batería debe

recargarse, con lo que el agente envía la orden al robot para que éste regrese, pudiéndose comprobar que el robot esta moviéndose hacia el recargador de batería.

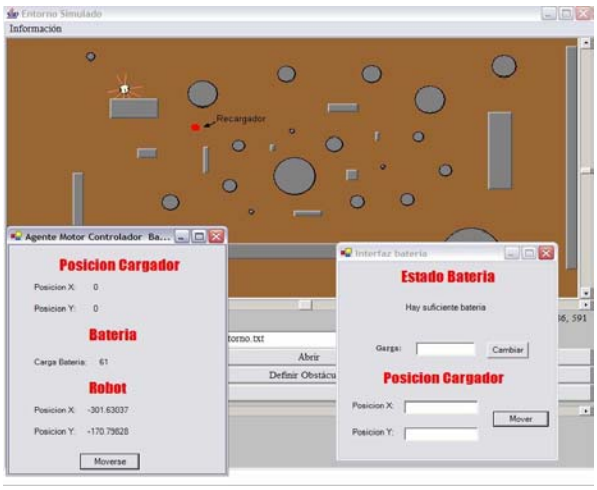

Figura 2: El robot cuenta con suficiente batería.

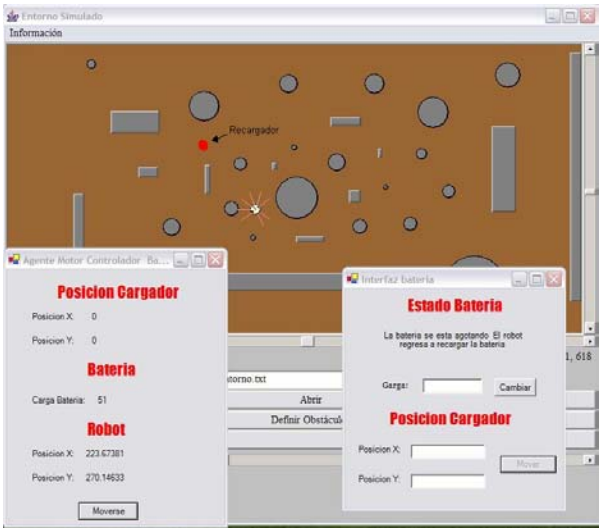

Figura 3: El robot debe regresar para cargar la batería.

En la última imagen (figura 4) puede observarse que el robot ha llegado donde se encuentra el recargador y esta llevando a cabo la recarga de la batería. Una vez recargada la batería el robot continuará con su funcionamiento habitual, tal y como se mostraba en la primera imagen (figura 2).

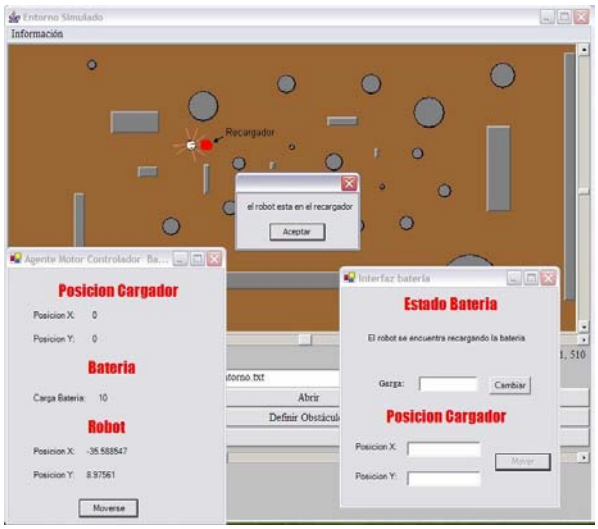

Figura 4: El robot se encuentra recargando la batería.

5 CONCLUSIONES

Se ha desarrollado una plantilla que puede ser usada como base para la creación de nuevos agentes *software* móviles basados en la arquitectura SC-Agent, proporcionando a los programadores una mayor facilidad y claridad a la hora de enfrentarse al reto que supone el desarrollo de un nuevo agente *software*.

Con esta plantilla el programador se despreocupa de todos los aspectos comunes que los agentes *software* comparten entre si (conexión con el sistema de comunicaciones, movilidad, etc.) y se centra únicamente en la parte de procesamiento que el agente a desarrollar va a tener que realizar para conseguir el objetivo para el que esta siendo programado.

Referencias

[1] Jennings, N.R., Wooldridge, M.J., (1998). Applications of Intelligent Agents. Agent Technology. Springer.

[2] Nii, H. P., (1989). Introduction, in: Blackboard architectures and applications. Edited by V. Jagannathan, Rajendra Dodhiawala and Lawrence S. Baum, (Perspectives in artificial intelligence, volume 3). Academic Press, Boston, pp. xix-xxix.

[3] Pérez, P., Posadas, J.L., Simó, J.E., Benet, G., Blanes, F., (2002). A Software Framework for Mobile Robot Sensor Fusion and Teleoperation. Proceedings of the 15th IFAC world Congress. Barcelona (España). I.S.B.N.: 008044184X.

[4] Posadas, J.L., Pérez, P., Simó, J.E., Benet, G., Blanes, F., (2002). Communications Structure for Sensory Data in Mobile Robots. Engineering Applications of Artificial Intelligence, vol. 15, no. 3, (2002) 341-350.

[5] Posadas, J.L., Simó, J.E., Poza, J.L., Pérez, P., Benet, G., Blanes, F., (2003). Una arquitectura para el Control de Robots Móviles mediante Delegación de Código y Sistemas Multiagente. IV Workshop de Agentes Físicos - WAF'03. Alicante (España). I.S.B.N.: 84-607-7171-7.

[6] Wooldridge, M., Jennings, N.R., (1995). Intelligent agents: theory and practice. The Knowledge Engineering Review, 10(2), 115-152.