Arquitectura software para robots móviles aplicando la metodología MDASR

Software architecture to mobile robots using the methodology MDASR

Nelson de J. Londoño Ospina. PhD.
Profesor Departamento de Ing. Eléctrica, y miembro del grupo de investigación en manejo eficiente de la energía GIMEL. Facultad de Ingeniería, Universidad de Antioquia, Medellín.Colombia.
nlondono@udea.edu.co

Recibido para revisión 24 de Abril de 2009, aceptado 23 de Octubre de 2009, versión final 25 de Noviembre de 2009

***Resumen*—Se aplica la Metodología de Desarrollo de Arquitecturas Software para Robots –MDASR-, como herramienta de especificación, análisis y diseño de una Arquitectura de Control Reactiva. La metodología se apoya en métodos y lenguajes ampliamente adoptados como el PU (Proceso Unificado de Desarrollo de Software) y UML (Lenguaje Unificado de Modelado), apropiando conceptos y métodos empleados en Ingeniería Software**

***Palabras Clave*—Robótica, Arquitectura de Robots, Ingeniería de Software, Proceso Unificado de Modelado.**

***Abstract*—Adopting a methodological strategy that covers the different stages of the development of a system, the Methodology of Development of Architectures Software for Robots - MDASR-, it is applied as tool of specification, analysis and design of a Reactive Control Architecture. The methodology rests on methods and languages widely adopted as the PU (Process Unified of Development of Software) and the UML (Language Unified of Shaped) and adapting concepts of methods used in Engineering Software.**

***Keywords*—Robotics, Robots architecture, Software engineering, Unified Modeled Process, software Development.**

I. INTRODUCCION

Tomando como base una propuesta Metodológica para Arquitecturas Robot [13], que cubre las diferentes etapas del desarrollo de un sistema, se presentan como caso de estudio la arquitectura reactiva aplicada a un sistema robot, siguiendo los pasos propuestos en MDASR (Metodología de Desarrollo de Arquitecturas Software para Robots [14]-[15], cuya característica más importante es que se apoya en métodos y lenguajes ampliamente aceptados por la comunidad científica y adoptados en diferentes áreas del conocimiento, como lo son: el PU (Proceso Unificado de Desarrollo de Software) [9] y el UML -Lenguaje Unificado de Modelado- [3]-[4]-[20]-[19], complementados con métodos empleados en Ingeniería Software [12][6].

Con base en la Arquitectura Reactiva aplicada a un robot comercial, se ejemplifica cada uno de los pasos propuestos en la metodología, enfatizando en los diagramas de casos de Uso, que permiten ilustrar la arquitectura, y presentando las etapas de análisis y diseño de forma general mediante los pasos concebidos en MDASR.

Es importante aclarar que, por limitaciones de espacio, todos los casos de uso, diagramas de diseño y análisis no pueden ser detallados, presentando solo algunos de los esquemas y resultados obtenidos en el desarrollo del proyecto.

II. MDASR - DIAGRAMA GENERAL

La Metodología de Desarrollo de Arquitecturas Software para Robots (MDASR) [14], es una metodología concebida para facilitar la comunicación entre el ingeniero en Robótica (quien concibe la arquitectura) y el ingeniero de sistemas (quien la desarrolla), se apoya en los paradigmas de la ingeniería de software, adoptando métodos de especificación, análisis, diseño e implementación estándares y el lenguaje de representación más relevante actualmente como es UML. La figura 1 ilustra, de forma gráfica, la metodología, que proponen los pasos a seguir en la concepción, análisis, diseño e implementación. Se toma como caso de estudio e ilustración de la metodología, una Arquitectura Reactiva aplicada a un robot comercial [10].

El objetivo es que, siguiendo los lineamientos de la metodología, la solución propuesta por el Ingeniero en Robótica, "Arquitectura Reactiva", sea representada de forma estándar,

guiada por un procedimiento sistemático, como se ilustra en la figura 1, para obtener un sistema completamente especificado por Casos de Uso, Analizado y Diseñado siguiendo un conjunto los pasos, bajo un lenguaje y procedimiento que llegue hasta el programador, quien en última instancia convertirá la arquitectura en un conjunto de elementos software.

En el ejemplo se toma cada paso y se desarrolla hasta un nivel que sea fácilmente comprendido por el ingeniero de sistemas

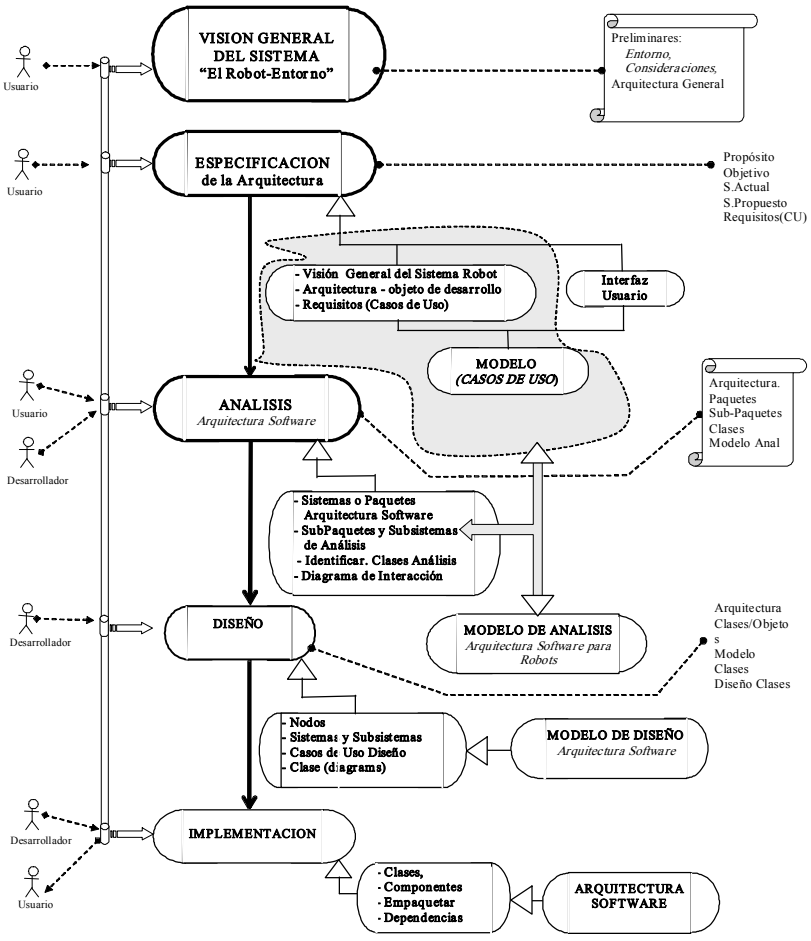


Figura 1. Diagrama general de MDASR.

III. ARQUITECTURA REACTIVA

En el campo de las arquitecturas software para robots, son muchas las propuestas enfoques y tendencias de control que se proponen. Es así como se cuenta con una gran cantidad de arquitecturas con orientaciones de acuerdo a las técnicas de procesamiento de información, algoritmos de evaluación de decisiones, etc. Unas de las mas recurrentes, cuando se trata de sistemas de navegación y exploración, son las arquitecturas reactivas [5]-[7]-[16]-[21]-[17]-[11]-[18], se basan en modelos biológicos y de acciones, que explican el comportamiento de diversos organismos vivos de tipo estímulo respuesta; así, el robot percibe el mundo y ejecuta, en respuesta, la acción que parezca más apropiada [17]. En este tipo de arquitecturas, la información percibida no se integra ni se asocia a ningún modelo del mundo.

Como se dijo anteriormente, para ilustrar la metodología, se seguirán los pasos propuestos por la MDASR, aplicados específicamente a una la arquitectura reactiva.

A. Especificación - Control Reactivo

El control reactivo, se concibe como una serie de módulos elementales, capaces de producir acciones de respuesta directa, en concordancia con la información sensorial disponible, este tipo de acción puede ser ejecutada como respuesta fija, ante un proceso de sensado (proceso reactivo o instintivo) o por ejecución refleja, después de ser aprendido (procesos reflexivos).

El modulo de control reactivo es el encargado de ejecutar los comportamientos elementales que requieran los módulos superiores, entendiéndose como comportamientos elementales aquellos conformados por un ciclo de sensorización, acciones elementales y combinación de ellas. Un comportamiento

elemental se define como un dispositivo computacional, que interpreta datos sensoriales, los procesa y produce una respuesta [8].

Visión general del sistema: El bloque Reactivo es el encargado de ejercer propiamente el control directo del Robot. Su objetivo fundamental es garantizar su navegación, operado por un sistema de Control Reactivo (CR), basado en Comportamientos [1]. Sus características principales son:

- Permitir ejecución de comportamientos elementales.
- Ser flexible, lo que permite incluir comportamientos adicionales.
- Permitir que su implementación soporte una gran variedad de técnicas, incluyendo reglas heurísticas, redes neuronales, lógica borrosa algoritmos genéticos, etc.
- Los comportamientos elementales son conmutados de acuerdo a los requerimientos del Usuario

Propósito del sistema CR: En general este bloque recibe las órdenes del USUARIO y garantiza su ejecución; para ello, toma la información sensórica requerida, la procesa, y envía una orden de Acción, resultado del procesamiento del Control Reactivo.

Alcance del sistema CR: En general, el control reactivo debe:

- Garantizar funcionamiento reactivo
- Velar por la seguridad
- Resolver problemas de atascamiento y errores producidos en el sistema (Software/Hardware).
- Informar las fallas ocasionadas a los niveles superiores.

Sistema actual. Se cuenta con el sistema Khepera básico, de consecución comercial [10].

Sistema Control reactivo propuesto. La Arquitectura Software propuesta, debe implementar el conjunto de operaciones que garanticen que el sistema ejecute control reactivo, el Caso de Uso principal es precisamente "Ejecutar_Control_ Reactivo", el cual ejecuta las acciones necesarias para garantizar dicho comportamiento y, por sus características, requiere un conjunto de comandos considerados Casos de Uso que extienden el principal. Por esta razón, es necesario realizar una especificación mas detallada de este sistema, que conlleve a definir un diagrama de casos de uso como se ilustra en la figura 2.

En el esquema propuesto, el actor puede ser un operador humano, un computador o simplemente un Caso de Uso de un sistema de control más general denominado "Ejecutar_Control _Reactivo".

Caso de Uso "Ejecutar Control Reactivo": La figura 3 propone el diagrama de Casos de Uso de una Arquitectura de "Control Reactivo".

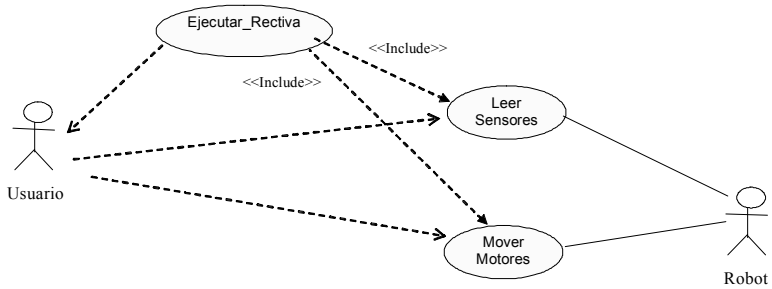


Figura 2. Caso de Uso "Ejecutar_Control_Reactivo"

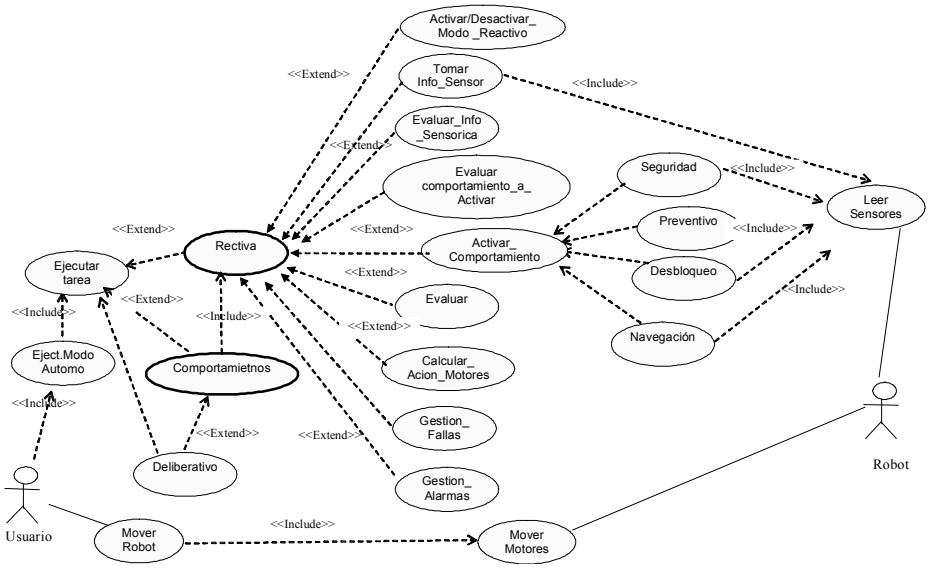


Figura 3. Diagrama de Casos de Uso detallado Ejecutar_Control_Reactivo"

Este tipo de diagramas, es producto de un proceso de especificación previo, que por limitaciones de espacio no se detalla en este artículo, en el que se han detallado y justificado los diferentes Casos de Uso que lo conforman y además, debe ir acompañado de una descripción, Identificación, Análisis y Refinamiento de cada uno de dichos Casos de Uso [15].

Desde una concepción general, "Control Reactivo" debe cumplir una serie de funciones que se resumen en el diagrama de casos de uso, pero que para efectos de claridad, en las primeras etapas de especificación, es conveniente definir con un poco mas de detalle. Por ejemplo, el control reactivo debe:

- Iniciar el modo de control Reactivo en el momento de recibir la orden correspondiente.
- Cargar comportamiento deseado por el usuario o definir comportamiento por defecto. El tipo de comportamiento está definido de acuerdo a las necesidades del usuario o en función de la respuesta en la evaluación de la información sensorial.
- Adquirir los datos de los sensores.
- Evaluar la información recibida. Implica definir el algoritmo de evaluación y los criterios de decisión definidos por el usuario
- Definir los comportamientos que deben se tenidos en cuenta de acuerdo a la misión y las condiciones de el medio (Información sensada).
- Activar el comportamiento.
- Evaluar el comportamiento activo y definir su continuidad o cambio de estado.
- Calcular la información de salida del comportamiento, como una acción de velocidad lineal y curvatura (Acción de Control), en parámetros normalizados y darlas ordenes a los motores.
- Ejecutar el algoritmo de gestión de fallos, cuando se detecte una condición no prevista.

También, se identifica, en la especificación del sistema reactivo esperado, una serie de requerimientos de Comportamiento (Behavioral), que ayudan a identificar, con mayor nivel de detalle, los elementos que definen el sistema de control deseado. Así, por ejemplo:

- El Software debe poder iniciar en cualquier instante, independiente del estado del Robot.
- La señal de Activar debe iniciar el proceso de comportamiento reactivo del robot e iniciar todas las tareas que se requieran. Por defecto arrancará en una configuración básica. Entonces, al iniciar:
- Habilita comportamiento Cargado.
- Calcula permanentemente la velocidad relativa Robot-Obstáculo
- Opera permanentemente en un modo protección (Prioridad).
- El robot permanece parado hasta tanto se de la orden de Arrancar.
- Debe permitir comportamientos adicionales, lo que exige modularidad y estandarización de los objetos.
- Debe permitir cambiar los comportamientos en el instante que lo requiera el usuario.
- Debe permitir una comunicación con el usuario, para informarlo del estado del sistema cuando lo requiera.
- En el instante que el usuario decide parar, todas las tareas que se estén ejecutando en el momento deben ser paradas y se debe dar orden de paro a los actuadores.
- Permite desactivar todo el sistema de control reactivo.
- Debe detectar las inconsistencias y fallos del sistema.

Análisis de Requerimientos no Funcionales: *Son* aquellas exigencias del sistema que no tienen efectos funcionales, pero que ayudan a identificar necesidades o restricciones, para el ejemplo:

- El sistema debe ser confiable, versátil y tener la posibilidad de interactuar con diferentes tipos de Robot o vehículos implemetado en una plataforma.
- El sistema de control obedece a un comportamiento reactivo, por lo cual su respuesta está sujeta a la velocidad de procesamiento, la simplicidad de los algoritmos y a la confiabilidad y sencillez de los sensores.
- Debe funcionar en diferentes tipos de ambiente, con una gran variedad de sensores y actuadores.
- Debe permitir incorporar otros módulos de comportamiento.
- Debe permitir involucrar, en los comportamientos, cualquier tipo de sistema de control: PID, Fuzzy, Neuronal, etc. (Comportamientos reflejos).
- Velar por la seguridad
- Resolver problemas de atascamiento y errores producidos en el sistema (Software/Hardware).
- Cada uno de los comportamientos puede ser considerado como una forma de Control del Robot.
- Su salida debe ser determinística
- Cada uno es completamente independiente
- Los comportamientos no poseen memoria.

B. Refinamiento Caso de Uso "Ejecutar Control Reactivo"

Una revisión general de Identificación y Análisis de Casos de Uso, implica detallar más la especificación, para muchos casos de uso, que deben ser iterativamente refinados, ejemplo: "Navegación", "Desbloqueo", "Preventivo ", "Seguridad", etc. Las figuras 4 ilustran el resultado del proceso de refinamientos del caso de uso "Navegación".

C. Especificación Comportamientos

Como se observa en la figura 4, otro elemento básico de una arquitectura de control, para un sistema robot, es el Caso de Uso Comportamiento, pues es posible que el robot tenga la capacidad de actuar, no solo en operación reactiva pura, sino obedeciendo a comportamientos [2]. El control de comportamientos del sistema, propuesto como objetivo básico del paquete, requiere ser detallado y, al igual que el anterior, debe ser especificado con mayor precisión. Producto de este procedimiento, se presenta el diagrama de Casos de Uso que implementa este paquete, figura 5. Como se observa, es dispendioso, pero necesario, realizar todos los procesos de identificación, análisis y refinamiento de cada uno de los casos de uso contemplados en el sistema.

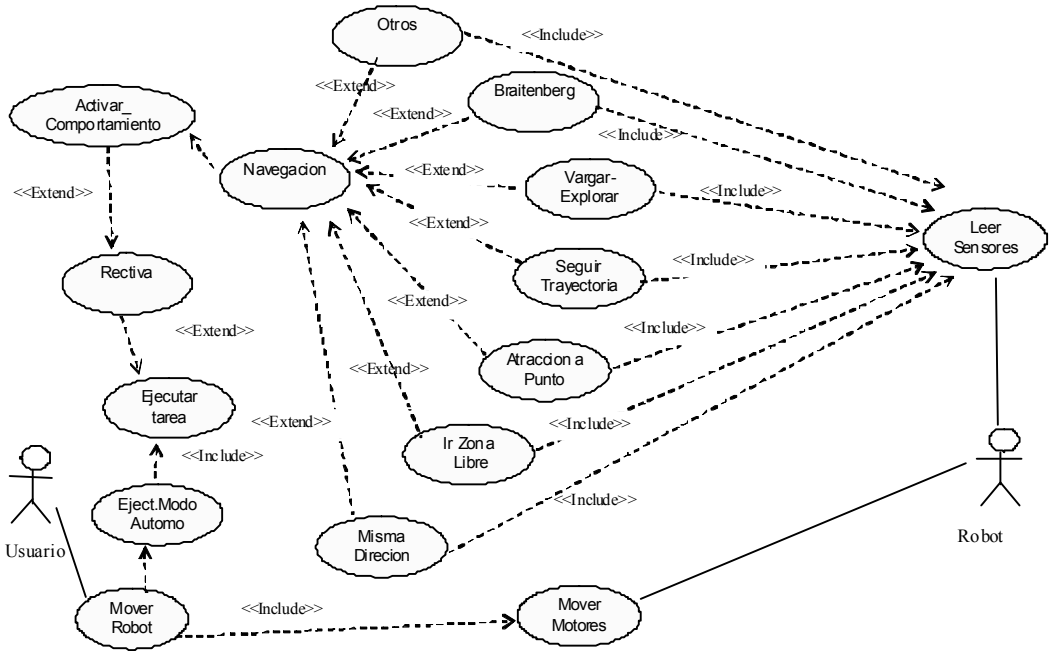


Figura 4. Diagrama de Casos de uso detallado "Navegación"

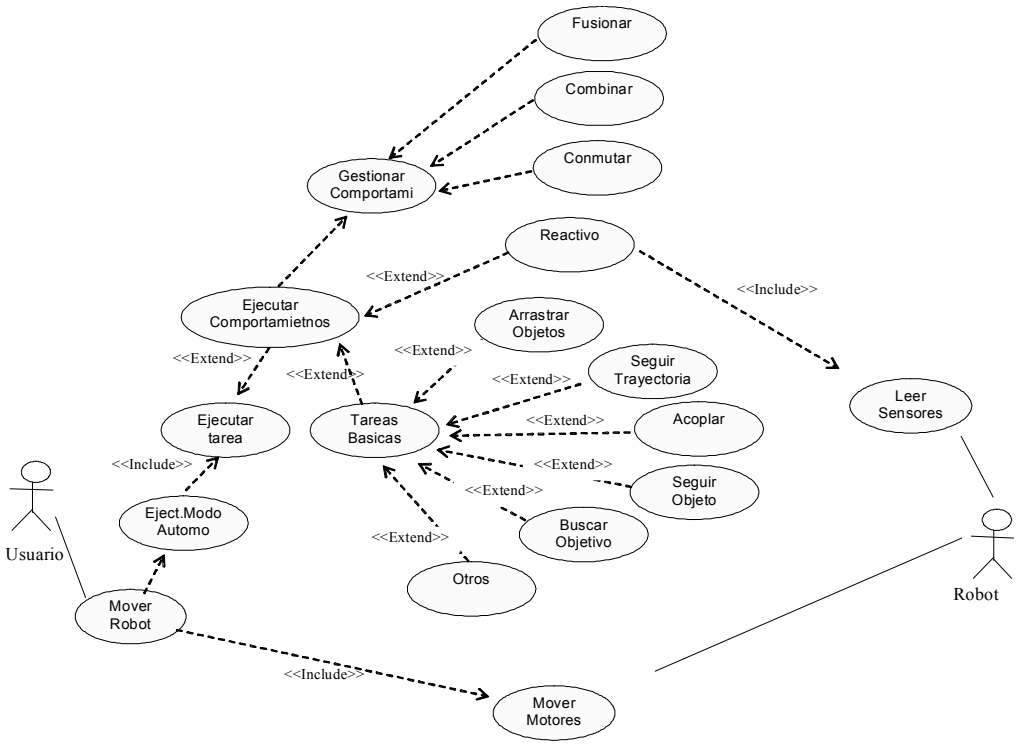


Figura 5. Diagrama de Casos de uso detallado que implementa "Ejecutar_Comportamientos"

D. Análisis - Identificación de Paquetes de Análisis -Por casos de uso

En este punto, cada uno de los paquete de análisis del sistema, va de la mano con el tipo de Caso de Uso que resuelven, así, por ejemplo , se define un paquete denominado "Sist_Navegación, que implementa todos los casos de uso relacionados con la navegación del robot en modo reactivo. En la figura 6 se ilustra el paquete y sus casos de uso que implementa.

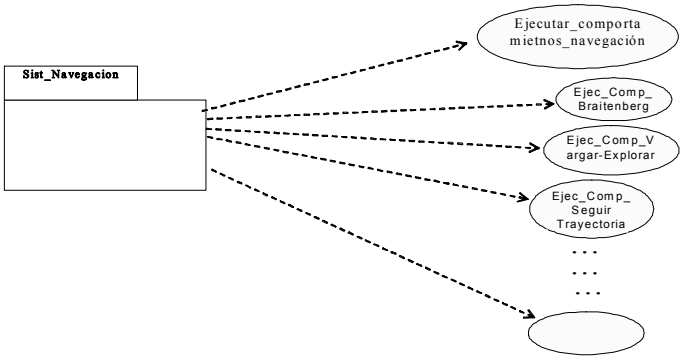


Figura 6. Diagrama de Casos de uso detallado que implementa "Sist_Navegación"

De forma similar, se plantean los otros paquetes del sistema de control reactivo que se resumen el numeral siguiente.

Arquitectura por paquetes: En el proceso de subdivisión del sistema software, el control reactivo puede ser clasificado de acuerdo a las características específicas de la función que cumple, en la Arquitectura propuesta. En nuestro caso, se subdivide dicho control en paquetes que desempeñan funciones específicas, en el contexto de la arquitectura y que, por sus características, implementan los casos de uso relacionados con su función.

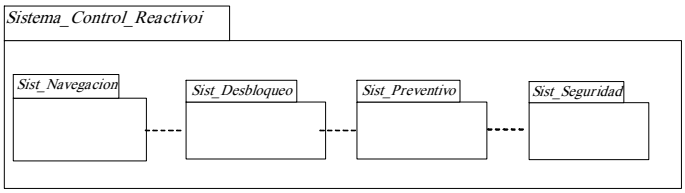


Figura 7. Arquitectura por paquetes de Sistema de Control Reactivo

En la figura 7 se presenta el sistema de control reactivo que cobija las cuatro funciones fundamentales de un sistema de control reactivo como el propuesto.

Cada uno de los sub-paquetes, contiene las clases y elementos necesarios para implementar las acciones correspondientes a su función específica.

E. Identificación de Clases

Tomando como base los paquetes y sub-paquetes, definidos en las últimas iteraciones del sistema de control anterior, se propone una serie de clases que implementarán cada uno de ellos.

A Partir de los (Sub) Paquetes de Análisis: Cada sub-paquete, en función de sus características y particularidades definidas en las etapas anteriores, puede ser implementado mediante un conjunto de clases que se identifican en esta etapa del proceso (en muchas ocasiones, están apoyados por el esquema general presentado por el ingeniero en robótica)

- Control Reactivo: El control reactivo, definido como un conjunto de paquetes que apoyan esta estructura de control, puede ser propuesto como un modelo de clases que implementan cada uno de estos paquete, en la figura 8 se ilustra algunos de estos paquetes, obtenidos mediante el procedimiento propuesto por la metodología.

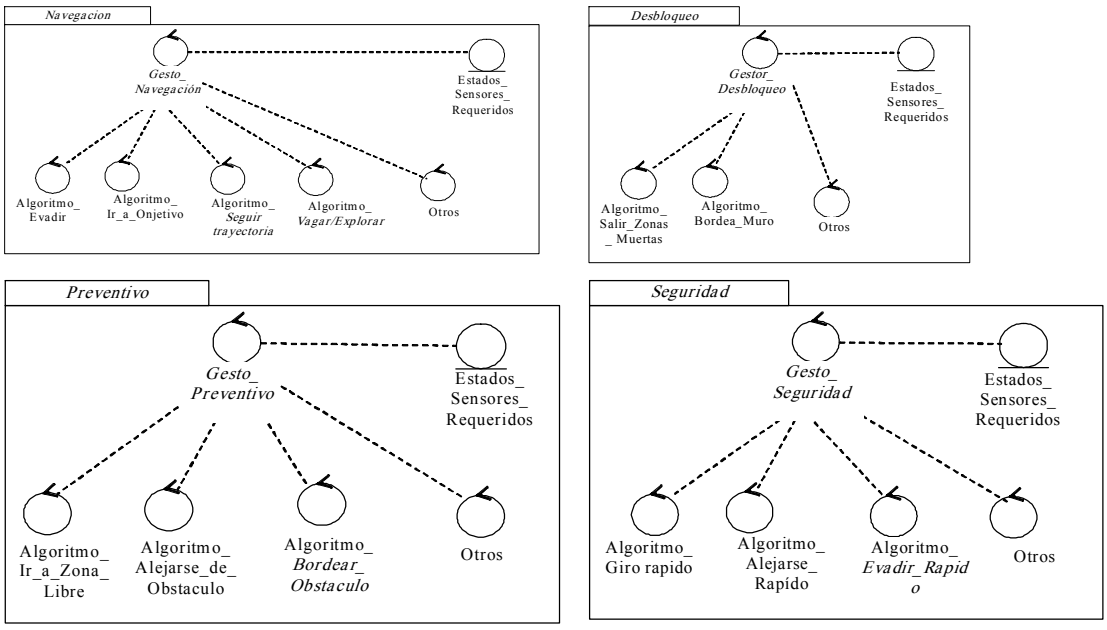


Figura 8. Modelo de Clases Sub-paquete: Navegación, Desbloqueo, Preventivo, Seguridad

A Partir de los Casos de Uso: En esta etapa, tomando como referencia los casos de uso definidos anteriormente, se identifican clases y su interrelación. En la figura 9, se ilustran algunos modelos de clases propuestos para la Arquitectura:

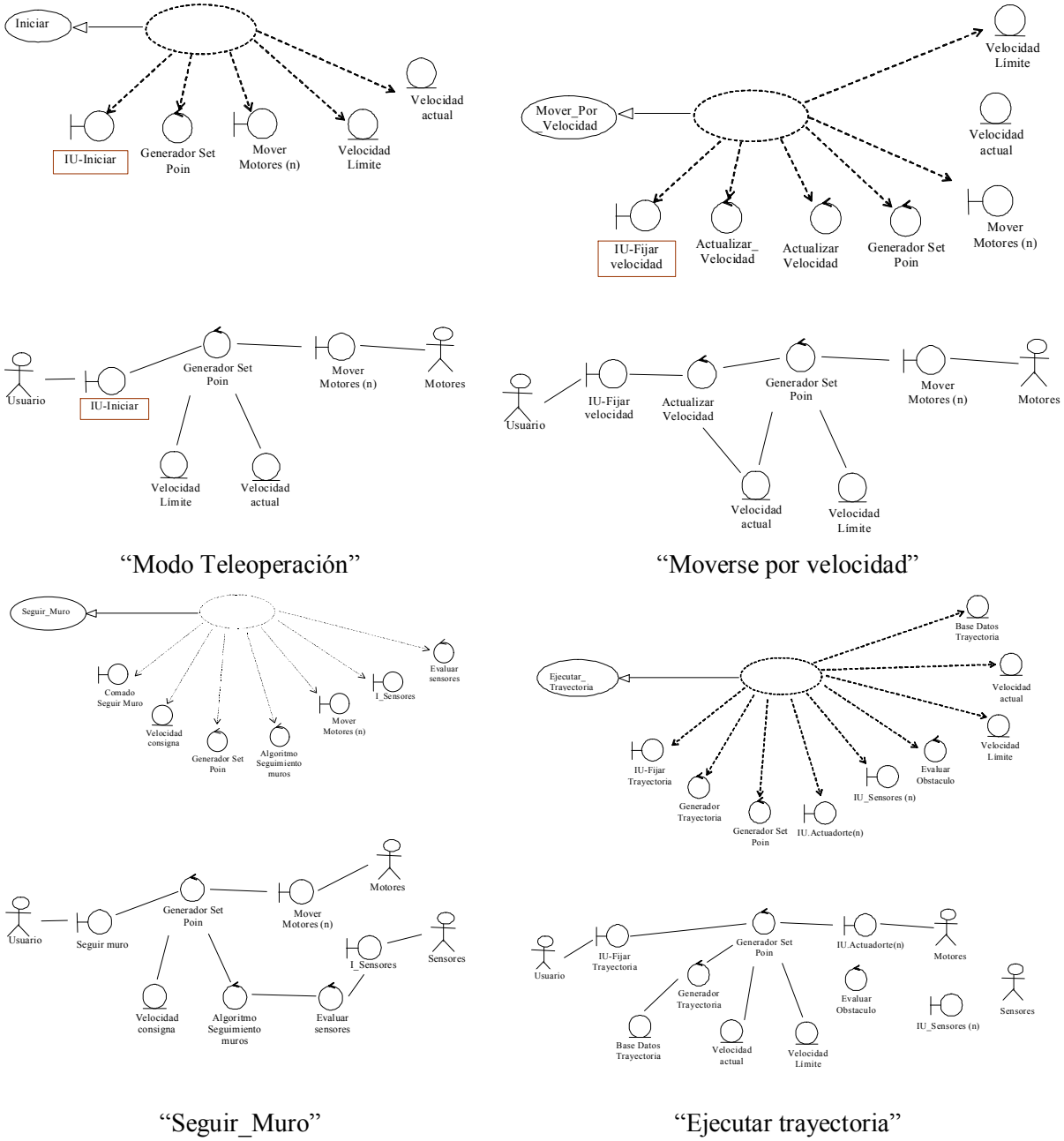


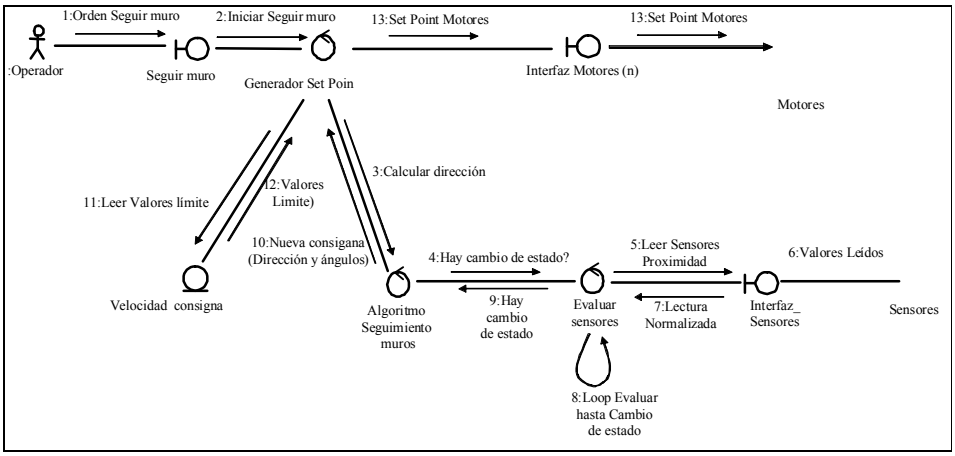
Figura 9. Identificación de clase e interacciones Caso de Uso

F. Interacciones Entre Objetos de la Arquitectura - Diagrama de Secuencias

El Diagrama de secuencias, correspondiente a cada uno de los casos de uso, se ejemplifica en la figura 10, en el cual se presenta solo el caso de condición normal y se omite la condición de bloqueo y colisión.

Caso de Uso "Seguimiento Muro" : Descripción textual del flujo de sucesos

- El usuario, mediante la interfaz (Gráfica, comando o botón), ejecuta la orden seguimiento de muro.
- El generador de Valor consigna (set point) evalúa sensores: si no se detectan obstáculos, genera un valor de consigna igual para los dos motores y lo envía a la interfaz Mover_Motores
- Si detecta obstáculos, sigue una secuencia de pasos definido por el flujo de eventos propuesto por la figura 10, que ilustra la relación entre clases que actúan es este caso de uso, y una propuesta inicial del flujo de eventos.



Flujo de sucesos - (Condición normal)

Figura 10. Interrelación y flujo de eventos. Caso de Uso "Seguir_Muro"

G. Modelo de Clases

El Modelo de Análisis de la Arquitectura de control Software para Robot, se presenta en la figura 11.

H. Diseño - Control Reactivo

El diseño está más próximo a los dominios del Ingeniero de Software y, por tanto, es éste quien, en última instancia, identifica y define cada uno de los elementos constitutivos de esta etapa. En esta sección se presenta, someramente, los conceptos más relevantes que han de considerarse, como ilustración en el marco del desarrollo de la arquitectura propuesta.

Refinamiento de la Solución: Tomamos como referencia el sistema de control reactivo. En la figura 12 se proponen los subsistemas que lo conforman, los cuales se subdividen de acuerdo a las características de comportamiento que debe ejecutar el robot, en función de su responsabilidad.

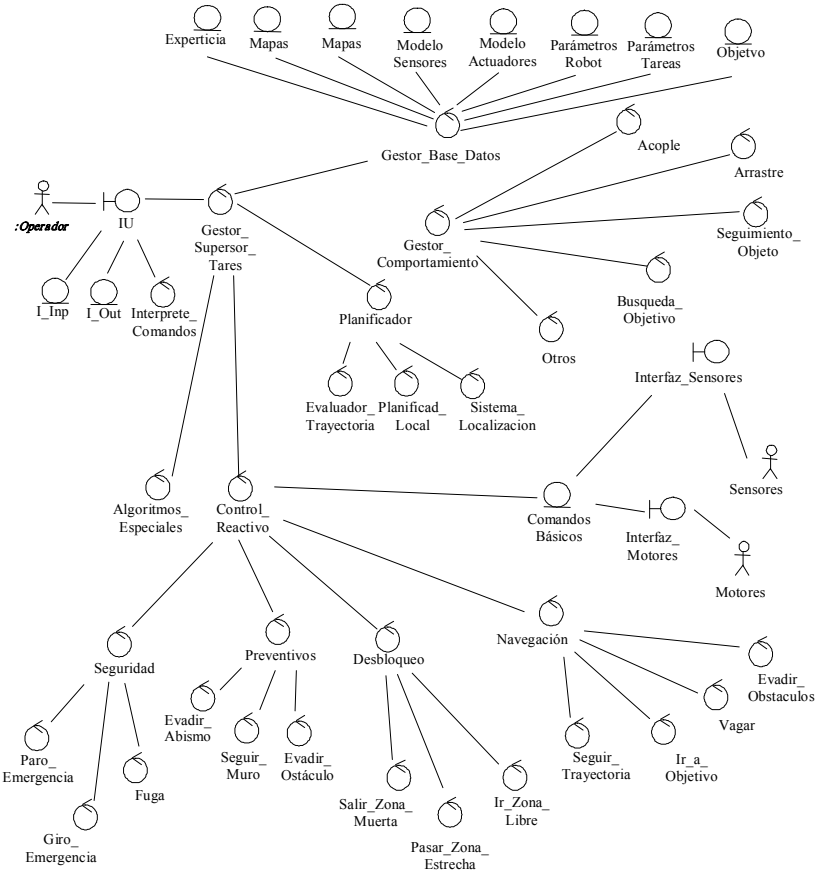


Figura 11. Ejemplo de Modelo de Clases de una Arquitectura Software de Control para Robots

Cada unos de estos subsistemas, a su vez, pueden ser subdivididos en unidades más pequeñas y manejables para facilitar la implementación y modularidad del software. Así, por ejemplo, el subsistema de navegación reactiva, puede ser subdividido en:

- Vagar (Navegación aleatoria)
- Seguir una trayectoria pintada en el suelo.
- Moverse hacia un objeto (Atracción a un punto).
- Ir a zonas libres dirigidas.
- Moverse en la misma dirección (permanecer en trayectoria).
- Arrastrar objeto.
- Moverse una dirección relativa (posición/Angulo)

A su vez, el sistema de navegación puede ser considerado como un subsistema (apoyado en el modelo de análisis), figura 13.

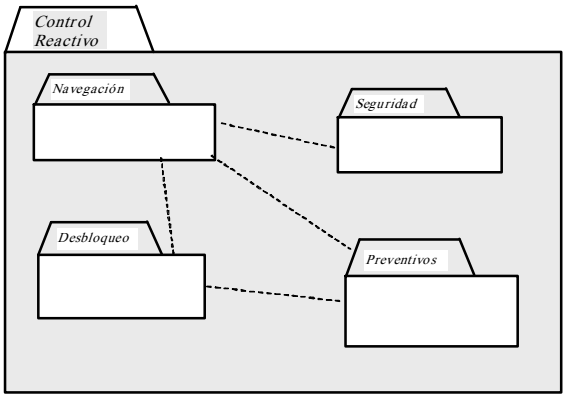


Figura 12. Subsistemas de "Control_Reactivo"

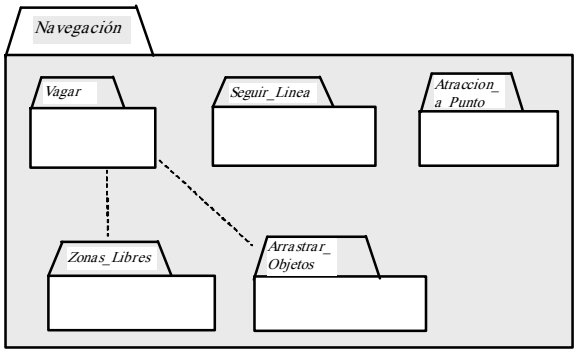


Figura 13. Subsistemas de "Navegación"

I. Identificación Interfaces de Subsistemas

La interfaz corresponde a una clase, que suministra todos los elementos necesarios para comunicar la clase con otras clases u objetos externos; por tanto, incluirá las características, atributos y métodos que permitan ejecutar la orden.

Nota: La relación entre subsistemas puede generar interfaces, así por ejemplo, para el Subsistema Control_Reactivo propuesto anteriormente, se puede representar sus diferentes interfaces como se ilustra en la figura 14.

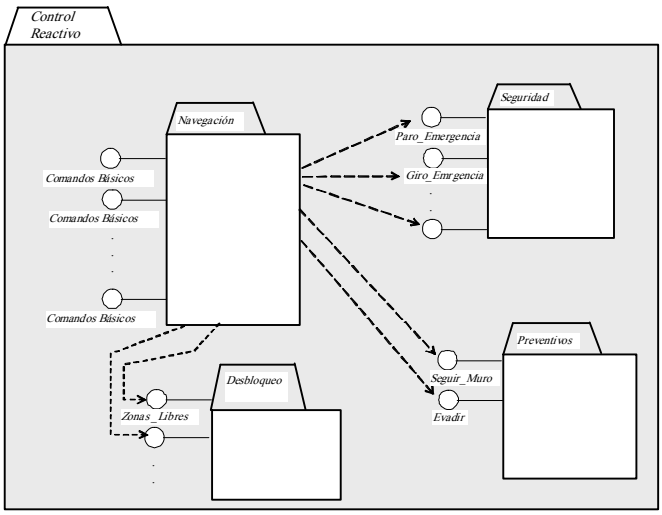


Figura 14. Interfaces "Relación entre Subsistemas y clases"

J. Diseño de Clases-Objetos

Diseño de Casos de Uso: Identificación - clases de diseño por caso de uso.

Tomando como base el Caso de Uso "Seguimiento de muro", visto en la etapa de análisis, se identifica cada una de las clases participantes en el Caso de Uso, y se presenta como un diagrama UML figura 15.

Diagrama de secuencia para cada parte de Caso de Uso: Para el Caso de Uso propuesto en el numeral anterior, se puede plantar el diagrama de secuencias, ejemplificado en la figura 16. Se asume, en este ejemplo, que los comandos se ejecutan vía teclado.

Nota: Podría considerarse que el usuario sea un nivel superior de control, en cuyo caso es posible que se requiera una interfaz que facilite la comunicación.

Clases de Diseño

Una versión simplificada del diagrama de clases para una arquitectura de robot se ilustra en la figura 17. Cada una de las clases representadas puede contener o extender otras clases.

Diseño de Cada Clases

El diseño de cada clase está ligado a las características y especificaciones definidas anteriormente. Como ilustración, se presentan algunos ejemplos representativos de este caso.

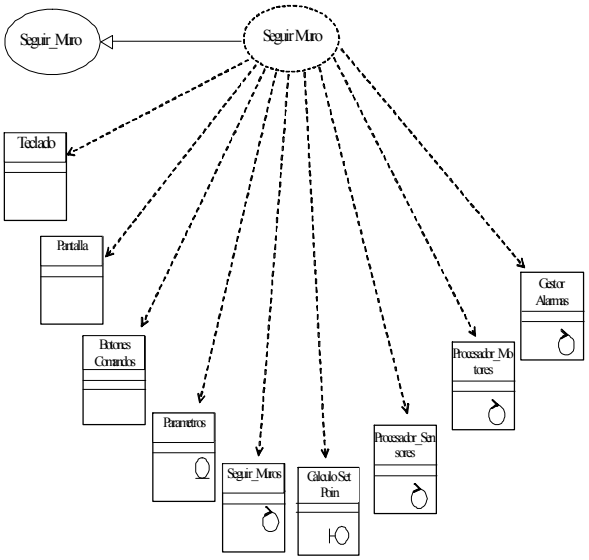


Figura 15. Clases que intervienen en el Caso de Uso "Seguir_Muro"

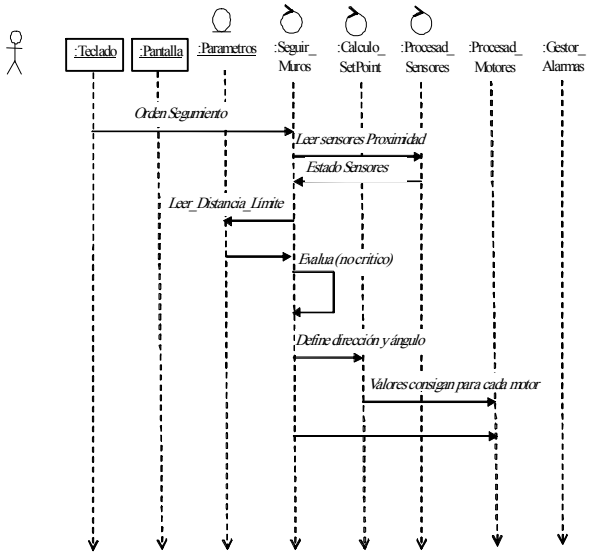


Figura 16. Diagrama de Secuencias, "Seguir_ Muro"

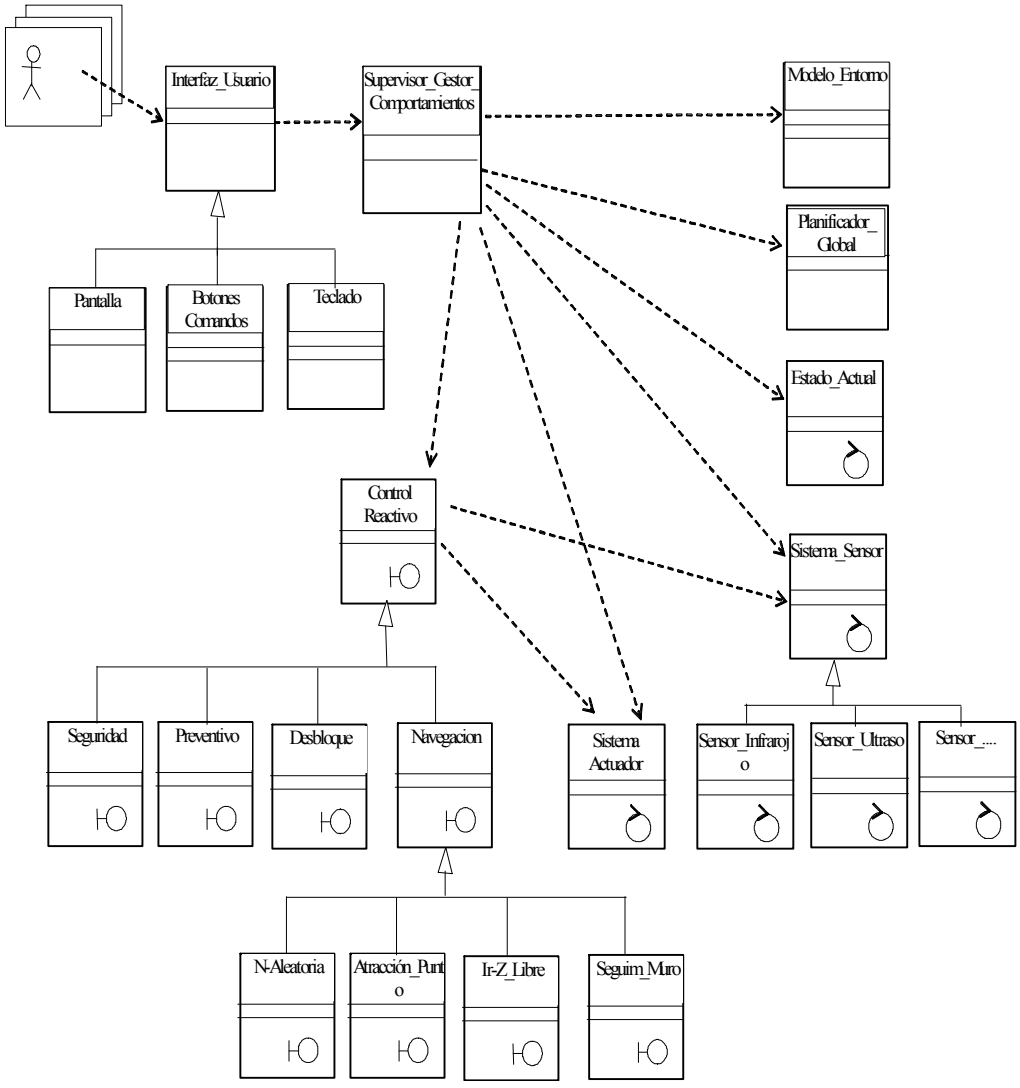


Figura 17. Ejemplo de Diagrama de Clases de Arquitectura Software para Robots

Clase: "Seguimiento_Muros"

- Esbozo de la clase: Seguimiento_Muros contiene los algoritmos necesarios para bordear los obstáculos que son detectados en el recorrido de robot, para ello utiliza como datos de entrada los valores captados por los sensores de ultrasonidos y la distancia a la que se realiza el seguimiento. A partir de esta información se genera como respuesta un vector (d,) que representa el incremento de distancia y ángulo que debe recorrer el robot, respectivamente. Cuando no se detecta obstáculo, la clase se inhabilita automáticamente.

- Esquema general: Figura 18

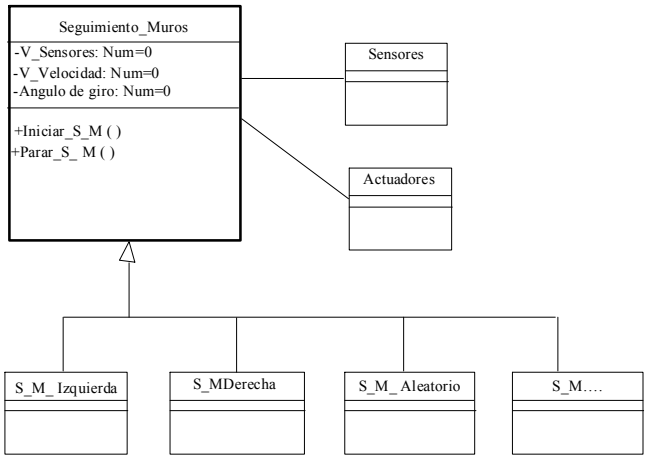


Figura 18. Representación gráfica de la clase "Seguir_Muro"

- Descripción de los métodos: En este aparte se define e método utilizado para el seguimiento de contorno que puede ser presentado como una formula o como un algoritmo, dependiendo del algoritmo escogido.
- Requisitos especiales: Para el seguimiento de contorno, la respuesta del algoritmo debe se menor que un tiempo predefinido por el usuario, o puede ser definida dependiendo de la velocidad actual del robot.

Los valores de ángulo y distancia pueden ser igualmente limitados a un valor mínimo para evitar daños en los dispositivos de actuación.

IV. CONCLUSIONES

Cubrir la brecha de comunicación entre disciplinas muy especializadas, fue el objetivo de la propuesta metodológica ejemplificada en este artículo, cuyo objetivo es, fundamentalmente, ilustrar los procedimientos mas relevantes al afrontar el desarrollo de una arquitectura de control reactiva, concebida por el especialista en Robótica y presentada de tal forma que sea fácilmente asimilada por el Ingeniero de Software, quien en última instancia es el que desarrolla el Software requerido.

En el artículo se hace una demostración un poco mas focalizada en el tema de la especificación y se ilustra, de manera general, los pasos relacionados con el Análisis y diseño de la arquitectura propuesta.

Se demuestra, que es posible, siguiendo un procedimiento sistemático, como lo propone MDARS, facilitar la concepción al Ingeniero en Robótica y su comunicación con el Ingeniero de Sistemas.

Como se observa, se hace un énfasis muy marcado el los Caso de Uso y su representación mediante Diagramas de Casos de Uso de la arquitectura propuesta, facilitando una aproximación muy rápida a las etapas de Análisis y Diseño.

Cubrir con más detalle las etapas de Diseño, Implementación, propuestas en la metodología (MDASR), al igual que una descripción más detallada de cada uno de los elementos software, no es objeto de este artículo pero, con él es posible identificar y evaluar los alcances y ventajas de la metodología, con el ejemplo propuesto.

REFERENCIAS

[1] Arkin, R. C. (www), 2009. http://www.cc.gatech.edu/ai/ robot-lab/ publications.html. [Con acceso en Feb. 2009].
[2] Arkin, R. C. 1999. Behavior-Based Robotics. 2ª ed. The MIT Press, Cambridge, Massachusetts, London, England.
[3] Booch, G., Jacobson, I., Runbaugh, J., 1999. El Lenguaje Unificado de Modelado. Addison Wesley. México.
[4] Booch, G., Rumbaugh, J.; Jacobson, I.,2005. The Unified Modeling Language User Guide" 2a Ed. Addison Wesley Professional.
[5] Braitenverg, V., 1984. Vehicles: Experiments in Synthetic Psycology". Cambridge. MIT Press.
[6] Brengge, B.; Dutoit, A. H., 2002. Ingeniería de Software Orientada a Objetos. Pearson Educación, México.
[7] Brooks, (www). http://www.ai.mit.edu/people/ brooks/ publications.shtml. [Con acceso en Feb. 2009].

[8] Gachet, D., 1993. Control Reactivo de Robots Móviles. Tesis Doctoral. Universidad Politécnica de Madrid.
[9] Jacobson, I.; Booch, G.; Rumbaugh, J., 2000. El Proceso Unificado de Desarrollo de Software. Adison Westley, Madrid-España.
[10] K-Team Corporation, 2009. Pagina disponible en internet en: <http://www.k-team.com>. [Con acceso en Feb. 2009].
[11] Lampe, A., Chatila, R., 2006. Performance Measure For The Evaluation of Mobile Robot Autonomy. Proceedings of the 2006 IEEE International Conference on Robotics and Automation Orlando, Florida.
[12] Londoño O., N., Tornero, J., 1998. Análisis de la Metodología HOOD para la Implementaciónde Arquitecturas Aplicadas a Robots. VIII Congreso Latinoamericano de Control Automático, Viña del Mar, Chile,
[13] Londoño N., 2009. Metodologías de Desarrollo de Software, un Enfoque a Robots Móviles.
Revista Politécnica. Politécnico Colombiano JIC.
[14] Londoño N., 2009. Aquitecturas Software de Robots, Metodología de Desarrollo. Octava Conferencia Iberoamericana en Sistemas, Cibernética e Informática. Orlando, Florida.
[15] Londoño N., 2009. Una Propuesta Metodológica para el Desarrollo de Arquitecturas Software Para Robots (MDASR). Tesis Doctoral, Universidad del Valle, Escuela de Ingeniería Eléctrica y Electrónica.
[16] Mataric, M. J.,1990. Behavior-Based Control: Main Properties and Implications". http://www robotics.usc.edu/~maja/publications.html. [Con acceso en Feb. 2009].
[17] Mukerjee, A., Mali, A. D., 1999. Reactive Robots and Amnesics: A Comparative Study in Memoryless Behavior. IEEE Transactions on Tystems, Man, and Cybernetics-part C: Applications and Reviews, Vol. 29, No. 2.
[18] Muñoz, N. D., 2007. Estudio e Implementación de una Arquitectura de Control para el Robot Móvil GIRAA_02. Tesis de Maestría. Universidad de Antioquia, Facultad de Ingeniería.
[19] OMG, 2003. Unified Modeling Language Specification. Retrieved March 18, 2003. Disponible en internet en: <http://www.omg.org/ technology/documents /formal/uml.htm
[20] Rumbaugh, J., Jacobson, I.; Booch, G., 2000. El lenguaje Unificado de Modelado, Manual de Referencia. Racional Software Corporation.
[21] Ram, A., Arkin, R. C., Moorman, K., Clark, R., 1996. Case- Base Reactive Navigation: A Method for on Line Selection and Adaptation of Reactive Robotic Control Parameters. IEEE, Transaction on systems, Man, and Cybernetics, Vol. 27, No. 3.