Un Ambiente para Programación y Control de Robots Móviles Autónomos

Erika Michalczewsky **Pablo R. Fillottrani**

Departamento de Ciencias de Computación
Universidad Nacional del Sur
Av. Alem 1253 – (8000) Bahía Blanca
Argentina

e-mail: {emichal,ccfillo}@criba.edu.ar

Resumen

Se presenta un ambiente para programación y control de robots móviles autónomos. El objetivo principal de este ambiente es proporcionar una herramienta que facilite el uso y la evaluación de las distintas técnicas existentes para planeamiento y control, simplificando la interacción entre los distintos módulos. Para ello se ha desarrollado una arquitectura basada en el sistema Saphira [Konolige 97a], que puede trabajar tanto sobre un robot real o una simulación del mismo provista por el sistema.

1 Introducción

La autonomía representa el principal beneficio de los sistemas de robots móviles. Esta característica permite a los robots concluir tareas útiles y además ser aceptados en cualquier ambiente abierto. Para cumplir con las tareas solicitadas por los usuarios, los programas de control del robot se encargan de adquirir información del entorno y actuar de acuerdo con su objetivo. El programa de control obtiene la entrada de los sensores, la procesa y decide las acciones motoras que ejecutará el robot inmediatamente. Sin embargo, este mapeo entre la entrada y la salida es muy complejo y requiere de la descomposición en elementos más sencillos.

Existen básicamente tres estrategias de control para los sistemas de robots móviles autónomos. Las arquitecturas de control centralizado tienen una componente que se encarga de recibir la información de los sensores, procesarla y decidir el próximo movimiento, basándose en las tareas pendientes, los obstáculos registrados, las nuevas tareas, etc. La desventaja de este tipo de arquitectura es que la responsabilidad de tomar todas las desiciones del sistema la tiene una única unidad. En el otro extremo, las arquitecturas descentralizadas tienen diferentes unidades que se responsabilizan por cada parte del procesamiento de la información, tomando decisiones que afectarán a las decisiones de las demás. En estas arquitecturas las componentes que reciben información de otras no pueden definir la razón por la cual se tomaron determinadas decisiones. Finalmente, en los últimos años ha habido una convergencia hacia una arquitectura híbrida para robots móviles autónomos. En la visión global de esta arquitectura se tienen tres elementos: uno en el nivel inferior, que corresponde al control de movimiento del robot; otro en el nivel medio, en el cual se encuentra el secuenciamiento de las acciones; y un último en el nivel superior, donde se realiza la planificación.

Se han presentado varias instancias de la arquitectura de control híbrida, incluyendo Saphira [Konolige 97a], ATLANTIS [Gat 92], SSS [Connell 92], Xavier [Simmons 97]. En estas arquitecturas, el secuenciador tiene el rol de ejecutivo principal. Esta componente considera las sugerencias del planificador e invoca los comportamientos para conseguir las metas requeridas. Cuando se escriben programas para robots, el resultado son programas del secuenciador. Para cada uno de estos ejemplos existen aplicaciones que trabajan mejor, y otras donde se tienen problemas. Es importante entonces tener un sistema que permita al usuario especificar interactivamente la estrategia de programación y control que considera adecuada para una situación, sin necesidad de crear un nuevo programa, compilarlo y ejecutarlo.

Actualmente existen sistemas que permiten manejar en cierto modo estas arquitecturas híbridas. El ambiente provisto por Saphira es un sistema integrado de sensado y control para aplicaciones de robots móviles, que opera en un ambiente cliente/servidor. El servidor tiene como labor controlar la operación de bajo nivel, abstrayendo al sistema de los detalles particulares del robot. Por su parte, el cliente Saphira controla las operaciones de alto nivel, desde el mismo robot o desde una computadora. Estas operaciones de alto nivel tienen como función controlar al robot interpretando la información de los sensores, utilizando comportamientos definidos con lógica difusa, realizando planificaciones y registrando los objetos del entorno.

El entorno en el cual se desplaza el robot junto con sus movimientos puede ser observado en el ambiente de desarrollo que contiene el sistema Saphira. Este ambiente ofrece un conjunto de herramientas para cargar archivos de programas, realizar la conexión y desconexión del servidor de robot, definir, iniciar, trazar y

terminar programas, etc. Sin embargo, el ambiente no ofrece herramientas que sean de fácil uso, como se verá más adelante.

El objetivo de este trabajo es presentar una herramienta que permita un manejo más sencillo y modular de las características de control de un robot móvil, utilizando como base el sistema Saphira. Este nuevo ambiente de trabajo presenta un escenario ideal para el aprendizaje, comparación y evaluación de las distintas técnicas de planificación y control. El nuevo sistema agrega nuevas funciones y modifica algunas características de la arquitectura del sistema Saphira. Entre ellas, se provee un editor de actividades dentro del ambiente, se permite la generación de mapas de manera interactiva, y otras que se describen más detalladamente en las próximas secciones.

El trabajo está estructurado de la siguiente forma: en la primera sección se detallan las características del sistema Saphira; en la segunda sección se evalúa su ambiente de desarrollo de aplicaciones, mostrando algunas deficiencias, luego se analiza la arquitectura del ambiente propuesto con el propósito de cumplir los objetivos mencionados y, finalmente, se describen algunos detalles de su implementación y los trabajos futuros a realizar.

2 El Sistema Saphira

Saphira fue desarrollado en el SRI International's Artificial Intelligence Center junto con el proyecto del robot móvil Flakey, como un sistema integrado para la percepción y accionar del robot. Este sistema opera en un ambiente cliente/servidor.

La librería de Saphira provee un conjunto de rutinas para construir los clientes, integrando funciones para enviar comandos al servidor, para reunir información de los sensores del robot y para preparar esta última para mostrarla en la interface gráfica. Además de las rutinas que realizan la tarea de comunicación y mantenimiento del servidor de robot, Saphira dispone de funciones de alto nivel para el control del robot y la interpretación de sensores, incluyendo comportamientos basados en control difuso, sistemas de planificación reactivos y un sistema de navegación y registración soportado por mapas.

El servidor de robot maneja los detalles de bajo nivel relacionados con los sensores y motores, se encarga de enviar información al cliente y responde a los comandos de Saphira a través de un protocolo de comunicación por paquetes. El cliente de Saphira se conecta al servidor de robot utilizando las componentes básicas para el sensado y navegación del robot: motores y ruedas, codificadores de posición y sensores.

Todas estas rutinas se encuentran organizadas en la arquitectura de Saphira, a través de dos sub-arquitecturas. La *arquitectura de sistema* consiste del conjunto de rutinas que permiten la comunicación y control del robot desde una computadora. Por encima de esta sub-arquitectura, la *arquitectura de control*, está diseñada con el objetivo de controlar los problemas que involucra la navegación, desde el control de los motores y sensores a la planificación y reconocimiento de objetos. Esta sub-arquitectura dispone de un conjunto de representaciones y rutinas para el procesamiento de la entrada de los sensores, para construir los modelos de mundos, y controlar las acciones del robot.

2.1 Arquitectura de Sistema

Saphira posee una arquitectura diseñada para operar con un *servidor de robot*, es decir, una plataforma de robot móvil que ofrezca un conjunto de servicios en un formato estándar. Esta plataforma es responsable de controlar la operación de bajo nivel de los motores por medio de comandos proporcionados por el cliente, de operar los sensores y de preparar los resultados de manera adecuada para retornarlos al cliente. Por encima de los controles de bajo nivel, el servidor implementa un conjunto de servicios básicos que incluyen:

- Control de movimiento: velocidad con dirección hacia adelante y hacia atrás y dirección angular. El servidor mantiene *setpoints* para controlar la velocidad y dirección. El cliente puede enviar comandos para modificar estos *setpoints*.
- Integración de posición: el servidor actualiza la posición del rumbo estimado del robot utilizando información de los codificadores del mismo.
- Sonar y otros sensores: el servidor coordina y dispara los sonares respetando una secuencia predeterminada. El cliente puede enviar comandos para modificar la planificación de los sensores.
- Comunicación: el servidor envía paquetes de información al cliente que contienen detalles acerca de la posición, velocidad y lectura de los sensores. También recibe comandos desde el cliente para actualizar sus variables y la planificación de sensores.

La *arquitectura de sistema* puede pensarse como el sistema básico para el control del robot. La figura 1 muestra la estructura de una aplicación de Saphira. Las rutinas de este sistema son microtareas que se encargan de la comunicación con el robot, de elaborar una imagen interna del estado del robot y tareas más complejas como la navegación e interpretación de los sensores. Estas microtareas son invocadas por el sistema operativo de microtareas que tiene Saphira.

2.1.1 Sistema Operativo de Microtareas

Las dos sub-arquitecturas que conforman Saphira están contruídas sobre un sistema operativo sincrónico, manejado por interrupciones. Las microtareas invocadas por este sistema operativo son máquinas de estado finito (MEF). El mismo cicla cada 100 ms a través de todas las MEF y ejecuta un paso en cada una de ellas. De esta manera, todas las MEF operan sincrónicamente ya que cada paso es ejecutado en intérvalos fijos de tiempo.

2.1.2 Rutinas de Usuario

Las rutinas de usuario son de dos tipos. El primer tipo es una *microtarea*, como las rutinas de las librerías de Saphira, que se ejecutan sincrónicamente cada 100 ms. Estas rutinas representan una extensión de las rutinas de las librerías y pueden acceder a cualquier nivel de la arquitectura de sistema. En general, el nivel más bajo

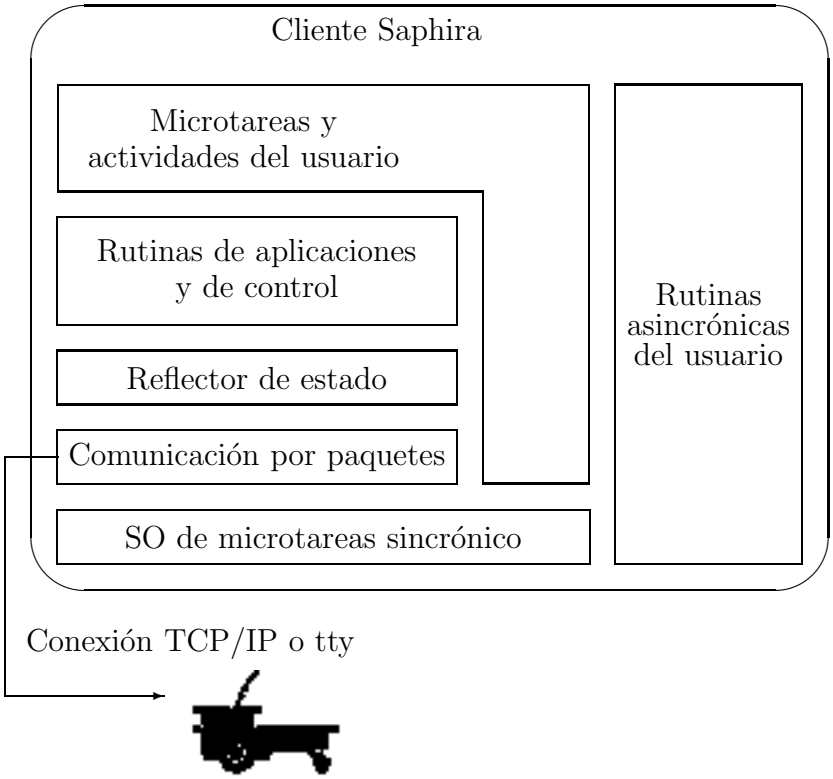

Figura 1: Arquitectura de Sistema

al cual pueden acceder es al *reflector de estado*, que consiste de una visión abstracta del estado interno del robot.

Todas las microtareas de Saphira y de usuario se encuentran escritas en el lenguaje C, y se ejecutan en el mismo hilo de ejecución, por lo tanto comparten las variables y las estructuras de datos. Las microtareas de usuario tienen acceso a toda la información utilizada por las rutinas de Saphira.

Las microtareas de usuario pueden ser codificadas directamente en C como MEF, sin embargo es mucho más conveniente escribir *actividades* utilizando el lenguaje Colbert, ya que se considera a las actividades como un tipo especial de microtareas.

Para tareas más complejas, como la planificación, para las cuales se requiere mayor tiempo, es necesario tener rutinas de otro tipo. Este segundo tipo de rutinas, *rutinas asincrónicas*, representan hilos de ejecución diferentes que comparten un espacio común de direcciones con las rutinas de las librerías de Saphira, pero son independientes del ciclo de 100 ms.

2.1.3 Comunicación por Paquetes

La comunicación en Saphira está soportada por medio del protocolo de comunicación por paquetes, a través del cuál se envían comandos al servidor de robot y se recibe información desde el robot. Los paquetes de información contienen las lecturas de sensores e información del movimiento de los motores. La conexión al servidor de robot puede realizarse utilizando una línea tty, una red Ethernet con TCP/IP, o un

vínculo local IPC.

2.1.4 Reflector de Estado

Saphira incorpora un *reflector de estado*, que muestra el estado del robot en la computadora para facilitar la tarea de los programas de control, de manera que estos no tengan que lidiar con el protocolo de comunicación. El reflector de estado muestra una visión abstracta del estado de los sensores y motor del robot. Los programas pueden recoger esta información y controlar al robot modificando los valores de control.

Es responsabilidad del sistema operativo Saphira de mantener el reflector de estado, a través de la comunicación con el robot, recibiendo los paquetes de información, actualizando el reflector de estado y enviando los paquetes de comandos para implementar esos valores de control.

2.2 Arquitectura de Control

La arquitectura de control de Saphira se encuentra por encima del reflector de estado (figura 2). Comprende un conjunto de rutinas que interpretan la lectura de los sensores de acuerdo con un modelo geométrico del mundo, y un conjunto de rutinas de acción que mapean los estados del robot en acciones de control. Las rutinas de registración vinculan las lecturas locales de los sensores al mapa del mundo, y el sistema de razonamiento procedural secuencia las acciones para lograr las metas específicas. La interface de agentes vincula al robot con otros agentes en una Arquitectura Abierta de Agentes. Un cliente típico utilizará sólo un subconjunto de esta funcionalidad.

2.2.1 Representación del Entorno

En el núcleo de la arquitectura de control se encuentra el Espacio Perceptual Local (LPS—Local Perceptual Space). El LPS es una representación geométrica del espacio alrededor del robot, y considera sólo unos pocos metros de radio teniendo en cuenta como centro al robot. Para un perspectiva más amplia, Saphira utiliza un Espacio de Mapa Global (GMS—Global Map Space) que permite representar objetos que son parte del entorno del robot, utilizando coordenadas absolutas (globales).

El LPS muestra al robot una percepción de su entorno local. Se utiliza para mantener control del movimiento del robot en intervalos cortos, de espacio y tiempo, recurriendo a las lecturas de los sensores y registrando los obstáculos que deben evitarse. El LPS está diseñado para acomodar varios niveles de interpretación de la información de los sensores, así como también de la información *a priori* originada a partir de mapas; puesto que distintas tareas pueden demandar diferentes representaciones. El LPS, además de brindar al robot conocimiento acerca de su entorno inmediato, es crítico en las tareas de fusionar la información de los sensores, planificar el movimiento local e integrar la información de los mapas.

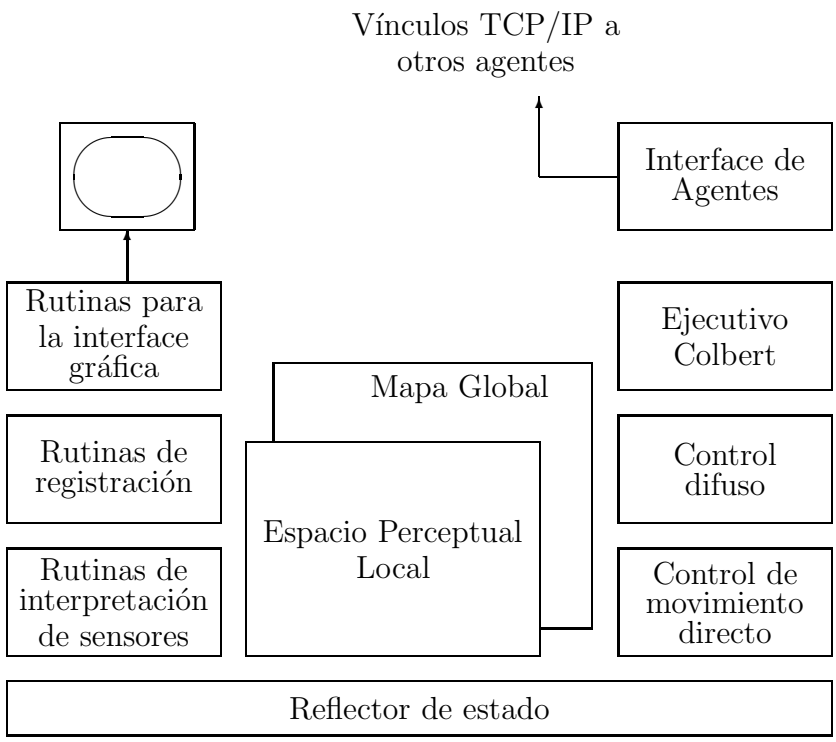

Figura 2: Arquitectura de Control

2.2.2 Control de Movimiento Directo

Es posible controlar el movimiento del robot de dos maneras diferentes. La manera más simple es modificar los *setpoints* de movimiento en el reflector de estado. Un *setpoint* de movimiento es el valor que el controlador de movimiento del robot trata de obtener para una variable de control. Al colocar un valor a una variable de control en el reflector de estado, las rutinas de comunicación reflejan este valor al robot, y sus controladores tratan de obtener este valor para la variable de control solicitada.

Los comandos de movimiento directo son apropiados para mover el robot a través de secuencias simples de acciones. Por ejemplo, utilizando las operaciones de **sfSetVelocity**, y **sfSetRVelocity** es posible modificar los valores de los parámetos de velocidad y velocidad de rotación, que se encuentran en el reflector de estado. Sin embargo, en ciertos casos la trayectoria del robot debe satisfacer las demandas complejas de las tareas y las políticas de mantenimiento.

2.2.3 Comportamientos y Control Difuso

Para el control de movimientos más complejos, Saphira soporta la facilidad que permite implementar *comportamientos*. Estos son escritos y combinados usando técnicas basadas en lógica difusa.

Cada comportamiento consiste de una *función de actualización* y un conjunto de reglas difusas. Las reglas de control mapean los estados del LPS en acciones de control para el robot. El propósito de la función de actualización es extraer

esa información del LPS y convertirla en un conjunto de variables difusas que sean apropiadas para el comportamiento. Dos de estas variables son la prioridad y el nivel de actividad, así como también otras variables que median la interacción con otros comportamientos y con las rutinas que los invocan. Las variables difusas son entradas para un conjunto de reglas de control difusas de la forma $A \rightarrow C$ donde A es una fórmula difusa compuesta por predicados difusos y los conectivos difusos AND, OR y NOT, y C es una acción de control.

Un robot puede tener varios canales de control, cada uno con la capacidad de realizar alguna función motora. Para cada canal de control, se promedian los resultados de todas las reglas de comportamiento para determinar el valor de control final. El promedio incluye una función de peso basada en dos propiedades: *prioridad* del comportamiento y *contexto* de activación.

2.2.4 Actividades y Colbert

Los comportamientos proporcionan el control de bajo nivel para las acciones físicas efectuadas por el sistema. Por encima de ese nivel, existe la necesidad de relacionar comportamientos a metas y objetivos específicos que el robot debe llevar a cabo. Este proceso involucra determinar cuándo activar/desactivar comportamientos como parte de la ejecución de una tarea, así como también coordinarlos con otras actividades del sistema. Para estas actividades, Saphira proporciona un método para planificar acciones del robot utilizando un nuevo lenguaje de control llamado *Colbert* [Konolige 97b]. Con este lenguaje se pueden construir librerías de actividades que secuencian acciones del robot en respuesta a condiciones del medio ambiente.

Los *esquemas de actividades* son los bloques básicos de Colbert. Cuando se instancian, el esquema de actividades es planificado por el ejecutivo de Colbert como otra microtarea, con funciones avanzadas para generar nuevas actividades y comportamientos, y coordinar acciones entre las actividades que se encuentran ejecutándose. El lenguaje tiene un rico conjunto de estructuras de control, y una sintaxis similar a C.

2.2.5 Rutinas de Interpretación de Sensores

Las lecturas actuales de todos los sonares se encuentran en estructuras en el reflector de estado, y a medida que el robot se mueve, las lecturas se acumulan en el LPS. Las rutinas de interpretación de los sensores se encargan de extraer los datos de los sensores o del LPS, y retornar información a este último.

Saphira activa los procesos de interpretación en respuesta a diferentes tareas: la detección de obstáculos, la reconstrucción de superficies, y el reconcimiento de objetos. Todas trabajan con los datos reflejados por los sonares y el sensado de movimiento. La interacción de la lectura e interpretación de los sensores que se lleva a cabo en el LPS permiten a Saphira coordinar sus nociones internas acerca del mundo con las impresiones de los sensores.

Más precisamente, su tarea consiste en analizar las secuencias de lecturas de los sonares y construir artefactos que corresponden a objetos del entorno, por ejemplo a través de las funciones **sfLeftWallHyp** y **sfRightWallHyp**. Otras rutinas tienen

como tarea determinar a partir de las lecturas crudas de los sensores si un obstáculo se encuentra cerca del robot. Estas funciones de detección tienen dos sentidos: las funciones *box* observan un región rectangular alrededor del robot, mientras que las funciones *plane* tienen en cuenta sólo una porción de una mitad del plano.

2.2.6 Registración de Objetos

Las estructuras en el GMS se denominan *artefactos*, y representan objetos del medio ambiente o estructuras internas, como por ejemplo vínculos. Una colección de objetos, como pasillos, puertas, y oficinas pueden ser agrupados en un mapa y guardados para ser utilizados más adelante. Estos mapas pueden ser creados por entrada directa de un archivo, o llevando al robot por el ambiente y permitiendo que Saphira extraiga la información relevante. Mediante un registro es posible mantener la ubicación global del robot en el ambiente en un mapa interno consistente con la lectura de los sensores. Existen además rutinas para extraer información relevante del LPS y compararla con las estructuras de los mapas en el GMS, y actualizar la posición del robot.

2.2.7 Mapas

Los comportamientos reactivos pueden considerar como argumentos de entrada directamente la lectura de sensores, con alguna transformación y filtrado. Por su parte, los comportamientos encauzados por metas se benefician con el uso de artefactos. Un artefacto puede representar una entidad artificial geométrica que guía el comportamiento. Los artefactos en Saphira se originan de tres maneras, por medio de información a priori, de características perceptuales e, indirectamente, de otros artefactos o información de las metas. Los artefactos son actualizados basados en el mecanismo de rumbo incierto. El funcionamiento de este mecanismo se sustenta en la coherencia, que es la propiedad de actualizar las posiciones de los artefactos en el LPS basándose en la percepción, de modo que el modelo en el robot acerca de su entorno permanezca registrado con su posición cuando se mueva.

3 Evaluación del Ambiente de Desarrollo

El sistema Saphira está acompañado de un ambiente de software con el cual se pueden programar y seguir los movimientos del robot[Konolige 98].

3.1 Descripción del ambiente

La ventana del ambiente Saphira contiene muchos de los elementos que se encuentran en el LPS, la cual puede apreciarse con el robot como centro, o en coordenadas globales. En la ventana se muestra el `ícono del robot`, la `lectura de los sonares`

mediante la acumulación de rectángulos pequeños, un `punto de control` que indica la dirección que debe mantener el robot, los `vectores de velocidad` que indican la velocidad de traslación y rotación del robot utilizando dos líneas que parten del centro del ícono del robot y por último las `áreas de sensibilidad de obstáculos` que son obstáculos dibujados temporalmente por los comportamientos de evitación de obstáculos.

El *área de información* contiene cuatro conjuntos de datos relacionados con el servidor de robot. El `estado (st)` muestra la condición del robot, por ejemplo `en movimiento`, `parado`, o `sin servo` cuando los motores se encuentran atascados. La `velocidad` está determinada por la velocidad traslacional (`Tr`) en milímetros por segundo y la velocidad rotacional (`Rot`) en grados por segundo. La posición(`X,Y,Th`) absoluta del robot se muestra en milímetros y grados. Los valores de comunicación en el área de información son el número de paquetes de un tipo dado recibidos en el último segundo. Normalmente, un cliente recibe 10 paquetes de motores (`Mpac`) y aproximadamente 25 paquetes de sonares (`Spac`) por segundo. Se muestran además otros datos como el nivel de voltaje de la batería (`Bat`) en el servidor, dato que indica si el robot necesita ser recargado.

En el *área de interacción*, Saphira imprime información acerca del sistema y el usuario puede tipear comandos para el evaluador de Colbert. Desde el área de interacción se pueden cargar los archivos de actividades y modificar el directorio de trabajo, realizar la conexión y desconexión del servidor de robot, definir, iniciar y terminar actividades, trazar actividades, obtener ayuda sobre las API y funciones del evaluador, y realizar un examen y fijar las variables internas de Saphira. El evaluador permite al usuario escribir y depurar programas desde el ambiente Saphira. Generalmente, el código del usuario se encuentra en un archivo de texto que contiene una mezcla de definiciones de esquemas de actividades y llamadas a funciones de las librerías. Durante la ejecución, el usuario puede examinar el estado de las variables de Saphira, iniciar y terminar otras actividades. Si ocurre un error, esa actividad se suspende y se imprime un mensaje. El usuario puede cambiar el texto del archivo Colbert, cargarlo nuevamente, y correr las actividades modificadas. No hay necesidad de terminar la aplicación y recompilarla.

La ventana principal tiene siete menúes. Estos menúes permiten controlar la información que se muestra en el LPS y en las sub-ventanas relacionadas, manejar la comunicación con el servidor y cargar los archivos de parámetros y de mapas.

3.2 Evaluación

El ambiente de programación de Saphira ofrece muchas facilidades que permiten ver y manejar los movimientos del robot real o del simulador. Sin embargo, todas estas herramientas requieren de un amplio conocimiento de la estructura de todo el sistema y el ambiente, incluyendo el lenguaje de control Colbert, necesario para programar las actividades.

A continuación se detallan algunas de las deficiencias que tiene el ambiente con respecto a su facilidad de uso y a la disponibilidad de las funciones.

Una de las primeras dificultades encontradas al revisar el sistema es la posibilidad de guardar el mapa generado por el robot durante su trayecto. Todos los datos acerca del mundo que son guardados en un mapa por el ambiente, a partir de la

información de los sonares, se mantienen mientras el cliente se está ejecutando, y se pierden una vez finalizada la aplicación. Este mapa mantiene datos acerca de los objetos reconocidos, como pasillos, puertas, y obstáculos en general. Si el ambiente en el cual el robot se mueve es siempre el mismo, es importante considerar la disponibilidad de esta función.

Para escribir actividades en el lenguaje Colbert debe utilizarse un editor de texto, no disponible en el ambiente. Esta es una facilidad sencilla de ofrecer para comodidad del programador.

Profundizando en la programación de un cliente Saphira, surge la necesidad de escribir nuevos comportamientos. El ambiente de Saphira no ofrece un modo sencillo que permita definir nuevos comportamientos. Estos deben ser especificados empleando una gramática particular para comportamientos, determinada por el sistema y que el programador del cliente debe entender para concretar esta tarea. El código escrito utilizando esta gramática, sirve como entrada a un compilador que devuelve otro código en el lenguaje C, el cual puede ser cargado, a través del evaluador de Colbert, al sistema Saphira. Este proceso no es natural, y además se encuentra complicado por la tarea misma de definir un comportamiento.

4 Arquitectura del Sistema Propuesto

El ambiente de desarrollo Saphira requiere del conocimiento de las componentes de su sistema para aprovechar sus beneficios. El objetivo del trabajo es obtener un nuevo ambiente, más amigable y con herramientas que permitan explotar las facilidades que ofrece Saphira, además de las que surgieron a partir de la evaluación del sistema y que fueron incorporadas. Este nuevo sistema utiliza varios de los bloques pertenecientes a la arquitectura de Saphira, además del simulador. Entre ellos el sistema operativo de microtareas, el protocolo de comunicación, el reflector de estado, etc. El bloque que conforma la interface gráfica es sobre el cual se realizaron las principales modificaciones, además de la incorporación de varias funciones a los bloques restantes.(ver figura 3)

En el ambiente Saphira, para cargar actividades el programador de clientes debe abrir un editor de texto cualquiera, escribir la secuencia de actividades que desea que realice el robot y grabarlas. Luego, una vez abierto el ambiente Saphira, debe establecerse una conexión con el servidor de robot (robot real o simulador). Si la conexión se estableció correctamente —según la información provista por las funciones de bajo nivel de Saphira —, se utiliza el área de interacción para cargar el archivo correspondiente a las actividades utilizando el comando `load`. La única manera de cargarlas es utilizando el área de texto. En el nuevo ambiente, es posible cargar las actividades desde el menú, además de poder editarlas en el mismo ambiente.

En el ambiente Saphira no es posible guardar el mapa actual en un archivo. Es decir, el modelo del mundo que generó el robot hasta un determinado momento en su recorrido por el entorno, no puede reternerse más allá de la terminación de la aplicación. En el nuevo ambiente esta facilidad de guardar el mapa permite al robot recordar nuevos objetos agregados a un determinado ambiente. Al mantener guardados los mapas de los ambientes, es posible recuperarlos más tarde sin la necesidad que el robot tenga que volver a reconocerlos.

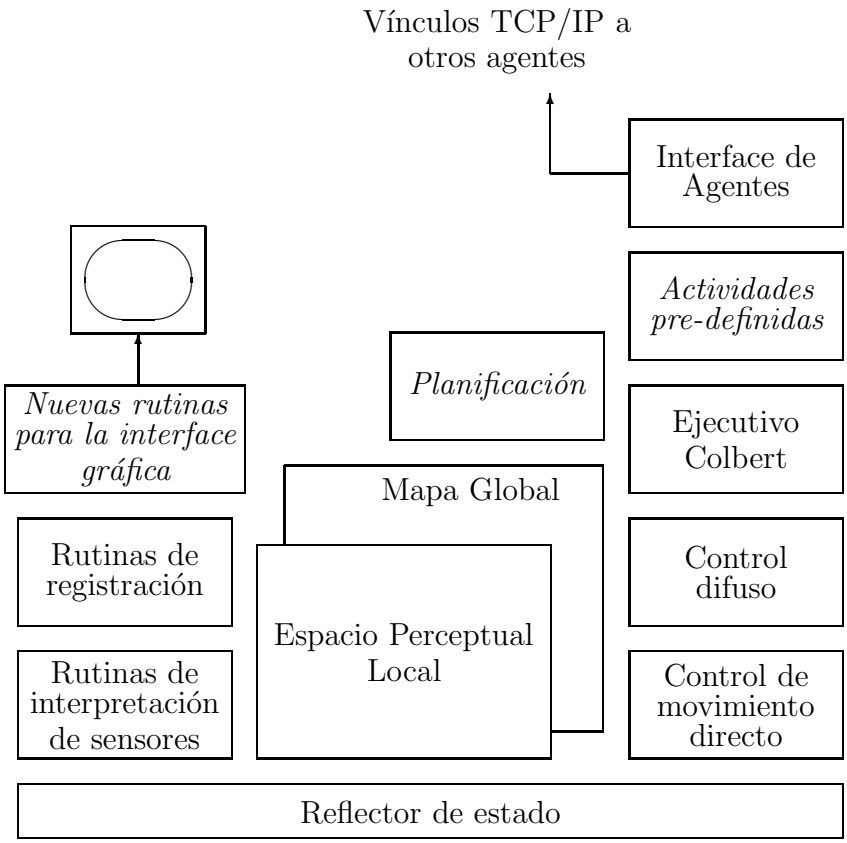

Figura 3: Nueva Arquitectura de Control

Saphira permite que el robot pueda cargar un mapa del mundo en el cual se desplazará para realizar sus actividades. Sin embargo, el robot no logra ubicarse dentro de ese mapa, generando así nuevos artefactos, sin hacer caso a los previamente localizados en ese mapa. La primera solución, y más simple, es tener alguna manera de indicarle al robot dónde se encuentra, para que pueda realizar una correspondencia más efectiva entre el mapa y el mundo. La segunda es buscar un algoritmo que realice una comparación más eficiente entre la lectura de los sensores y el mapa cargado. El nuevo ambiente permite comparar las estructuras de datos que representan a los artefactos del mundo con la lectura de los sensores, facilitando así la tarea de ubicación del robot.

Por otra parte, Saphira ofrece la facilidad de que el robot se encargue de reconocer el lugar y construya su propio mapa. Sin embargo, para el objetivo de este trabajo no es necesario que el robot realice este reconocimiento. En su lugar, el propósito es solicitar al robot el cumplimiento de una determinada meta habiendo cargado previamente los datos acerca del lugar en el cual deberá moverse. Además, es importante que el robot consiga una planificación adecuada para todas las tareas requeridas. Una vez cargado el mapa, y ubicado el robot, existen un grupo de tareas que el robot debe planificar para llevarlas a cabo de manera eficiente. En esta planificación el robot deberá tener en cuenta aspectos, como por ejemplo, que el camino más corto no siempre es el más conveniente. Más precisamente, si en

el camino más corto existe alguna puerta que debe atravesarse y generalmente esa puerta se encuentra cerrada, probablemente un camino alternativo, aunque más largo sería una mejor opción. Utilizar caminos que estén delineados con finales de pasillos son también una mejor opción para la planificación, que la posibilidad de caer en pasillos sin salida. Otra consideración es que al tener identificadas las oficinas o cuartos, el robot determina el orden en que las visita, tratando de aprovechar la cercanía de las mismas.

Para establecer metas interesantes en el funcionamiento del robot debe escribirse código complejo en el lenguaje Colbert. Como uno de los objetivos del trabajo es la sencillez del ambiente, se propuso disponer de un modo más sencillo de plantear las actividades. Es decir, la incorporación de actividades previstas que puedan ser agregadas seleccionándolas desde el menú. Se está trabajando en la posibilidad de que se puedan incorporar actividades dinámicamente.

Saphira alcanza metas complejas a un costo muy grande que es el trabajo de programación que debe realizarse en Colbert. Es necesario, modificar la estructura del sistema para agregar un manejo más abstracto de la planificación de tareas. Este elemento debe estar acompañado con la sencillez para definir las tareas y del mapa en el cual el robot tenga identificado únivocamente los lugares a recorrer. Así, la especificación de tareas puede hacerse de manera más abstracta.

5 Implementación y Trabajos Futuros

En este momento se está trabajando en la implementación del ambiente descripto en la sección anterior, manteniendo varias características del ambiente original y agregando la posibilidad de incorporar funciones arbitrarias a través de librerías dinámicas, que proporcionarían las distintas estrategias de planificación y control.

En este nuevo ambiente se incorporó el editor de texto, que permite además cargar los archivos desde el menú. Trabajar con este editor facilita la tarea de definir las actividades porque que es posible seleccionarlas desde el menú y se le evita al programador tener que especificarlas en Colbert. Los mapas generados por el robot durante su trayecto pueden ser guardados por el sistema, permitiendo su utilización nuevamente. Además, se incoporó a la arquitectura un bloque de planificación de las actividades, que permite un mejor ordenamiento de las tareas pendientes.

Todo este nuevo conjunto de herramientas ofrece un escenario en el cual es posible programar actividades para el robot de manera mucho más sencilla que en el ambiente de Saphira. Como ambiente de programación dispone de la facilidad para incorporarse como instrumento de estudio para el aprendizaje de técnicas de planificación y control en sistemas de robots móviles autónomos. Además, desde el punto de vista práctico es posible integrarlo como un participante más en un ambiente de oficinas, permitiéndole realizar tareas como la entrega de correspondencias, mensajes, faxes, etc.

La implementación del ambiente se está realizando en el lenguaje C++, aprovechando las librerías de Saphira disponibles. Se tiene previsto implementar una versión en un applet en Java con el objetivo de permitir la programación y control del robot a través de una red de computadoras.

Una vez implementado este nuevo ambiente de desarrollo, se tiene previsto uti-

lizarlo como herramienta didáctica en varios cursos relacionados. De la evaluación que surja se podrán diseñar futuras versiones no necesariamente dependientes de las funciones de bajo nivel de Saphira. La arquitectura del sistema presenta la flexibilidad que se necesita para lograr este objetivo.

Referencias

[Arkin 98] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[Connell 92] J. Connell. *SSS: A hybird Architecture applied to robot navigation*, en *Proceedings of the IEEE Conference on Robotics and Automation*, 1992.

[Gat 92] E. Gat. *Integrating planning and reacting in a heterogenous asynchronous architecture for controlling real-world mobile robots*, en *Proceedings of the AAAI Conference*, 1992.

[Konolige 98] Kurt Konolige. *Saphira Software Manual, version 6.1*, 1998.

[Konolige 97a] Kurt Konolige, Karen Myers, Alessandro Saffiotti y Enrique Ruspini. *The Saphira Architecture: A design for autonomy*, *Journal of Experimental and Theoretical Artificial Intelligence*, 1997.

[Konolige 97b] Kurt Konolige. *COLBERT: A Language for Reactive Control in Saphira*, 1997.

[Myers 96] Karen L. Myers. *A procedural knowledge approach to task-level control*, en: Proceedings of the Third International Conference on AI Planning Systems, AAAI Press, 1996.

[Simmons 97] Reid G. Simmons, Richard Goodwin, Karen Zita Haigh, Sven Koenig, Joseph O´Sullivan and Manuela M. Veloso. *Xavier: Experience with a Layered Robot Architecture*, *ACM Sigart Bulletin*, 1997.