Revisión de las arquitecturas de control distribuido

Autores:

José Luis Poza Luján

Juan Luis Posadas Yagüe

Revisor:

José Enrique Simó Ten

Instituto de Automática e Informática Industrial

(ai2)

Universidad Politécnica de Valencia

(UPV)

Versión: 1.0

Fecha revisión: 30 de marzo de 2011

Contenidos

Figuras

Tablas

Ecuaciones

1 Introducción

1.1 Resumen

Una de las claves en el control de sistemas es la arquitectura escogida para implementar dicho control. La elección de la arquitectura o el diseño de la misma determinarán, en gran medida el rendimiento que el control proporcionará al usuario. Existe una gran cantidad de arquitecturas de control en todos los ámbitos que éste cubre. Por ello parece conveniente realizar una revisión y exposición de las mismas, ya que de ésta manera se dispondrá de información suficiente para poder diseñar una arquitectura con las características más adecuadas a las funciones requeridas.

En el presente documento se realiza una revisión exhaustiva de diferentes arquitecturas en dos de los ámbitos del control: la domótica y la navegación de robots. Estos dos ámbitos, que a primera vista parecen lejanos están, en parte, relacionados ya que cubren todos los ámbitos de las necesidades de control temporal, desde los bajos requerimientos de la domótica hasta las necesidades de tiempo real estricto de la navegación reactiva de robots.

En el documento se hace especial hincapié en las características que las arquitecturas de control deben tener, ya que seleccionar correctamente las características de los requerimientos del sistema a controlar permitirá seleccionar o diseñar correctamente la arquitectura de un sistema.

1.2 Historial de revisiones

Nº revisión	Fecha	Comentarios
0.0	2006-10	Inicio del documento
0.1	2007-12	Final revisión de arquitecturas de navegación de robots
0.2	2008-01	Inclusión de las arquitecturas de control domótico
0.3	2008-02	Inclusión de las revisiones de características de las arquitecturas de navegación de robots.
0.4	2008-06	Final revisión de arquitecturas de control domótico
0.5	2008-07	Inclusión de las revisiones de características de las arquitecturas de control domótico.
0.6	2008-09	Modificación en introducción y conclusiones
0.7	2008-11	Inclusión de anexos con arquitecturas y software revisado.
0.8	2009-02	Inclusión del capítulo genérico de arquitecturas de control.
0.9	2009-04	Inclusión de la optimización de sistemas de control
1.0	2009-07	Revisión global del documento.

1.3 Objetivos del documento

El objetivo principal de éste documento es revisar diversas arquitecturas de control que se han publicado, abarcando el mayor número de tipos posible, para poder contextualizarlas en el ámbito del control y extraer así las características más deseables en un sistema de control eficiente. Para ello se deberán alcanzar diversos objetivos parciales:

- Revisar diversas arquitecturas domóticas, como ejemplo de arquitecturas con requerimientos bajos de control temporal, para extraer las características generales de éste tipo de sistemas.
- Revisar diversas arquitecturas de navegación de robots, como ejemplo de arquitecturas con requerimientos temporales medios y estrictos, para poder contextualizarlas dentro de los sistemas de control.
- Extraer las características más relevantes de las arquitecturas de control

1.4 Alcance y audiencia

Este documento cubre las arquitecturas de control domótico y de navegación de robots. El criterio de elección ha sido diferente entre las arquitecturas domóticas, donde el escaso número ha hecho que se hayan revisado todas las conocidas, y las arquitecturas de navegación, donde se han escogido las arquitecturas más extendidas y las más representativas dentro de alguno de los paradigmas.

El documento es de utilidad a aquellas personas que deseen conocer las diferentes arquitecturas pasadas y actuales en el ámbito del control domótico y de navegación de robots.

Asimismo también está dirigido a investigadores que deseen conocer las características más relevantes de las diferentes arquitecturas de control domótico y de navegación de control, especialmente si deben realizar el diseño de una arquitectura para un sistema de características similares.

1.5 Organización del documento

El documento se organiza de la siguiente forma. En el capítulo 2 se definen las arquitecturas de control y se contextualizan las arquitecturas revisadas. En el capítulo 3 se revisan las arquitecturas domóticas, inicialmente se repasan las características de los sistemas domóticos, para a continuación revisar ocho arquitecturas. En el capítulo 4 se sigue un esquema similar al capítulo anterior pero con las arquitecturas de navegación de robots. Finalmente se exponen algunas conclusiones acerca de las arquitecturas expuestas, así como la propuesta de características de optimización de arquitecturas y posibles líneas de investigación en el campo de las arquitecturas de control.

2 Arquitecturas de control

2.1 Definición

La arquitectura de un sistema se define como la organización fundamental de un sistema, que incluye sus componentes, las relaciones entre sí y el ambiente, y los principios que gobiernan su diseño y evolución [IEEE, 2000]. Desde un punto de vista más práctico, una arquitectura se puede entender como un concepto abstracto que permite describir un sistema, lo que se hace desde un punto de vista estático, y su funcionamiento, lo que se lleva a cabo desde un punto de vista dinámico.

La concreción de componentes que forman la arquitectura va aumentando a medida que ésta cierra más su ámbito de actuación desde los sencillos sistemas domóticos hasta sistemas más complejos como la navegación de robots móviles. Los componentes más comunes (figura 1), aunque dependientes del tipo de control que se desee realizar, suelen ser los componentes responsables de la adquisición de la información, procesamiento de la misma, la toma de decisiones acerca de las acciones de control que el sistema debe realizar y la gestión de las tareas requeridas por las acciones.

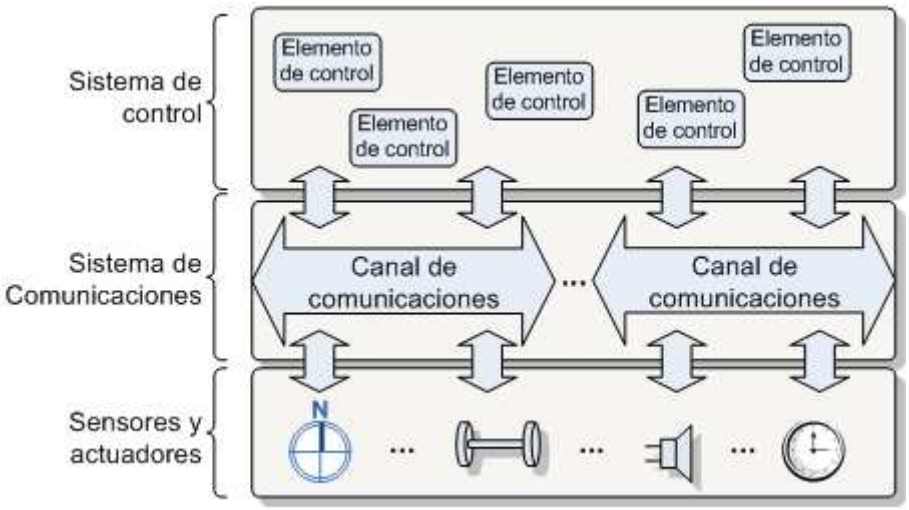

Figura 1. Elementos básicos de una arquitectura de control distribuido.

A medida que se desea un control autónomo, los componentes de las arquitecturas de control pueden coincidir con componentes de arquitecturas dedicadas a otros ámbitos, este es el motivo de la gran relación entre las arquitecturas de control y las arquitecturas de sistemas de inteligencia artificial. Esta relación tan estrecha, hace que las implementaciones de las arquitecturas también sean muy similares, por ello es normal encontrar implementaciones de sistemas de control de sistemas distribuidos basadas en arquitecturas de agentes empleados en la resolución de problemas de inteligencia artificial. Esto último se debe fundamentalmente a que el control de sistemas inteligentes, especialmente los distribuidos, son muy útiles como banco de ensayos de los sistemas de inteligencia artificial.

Existen muchas definiciones del concepto de control de sistema, básicamente se habla del proceso por el que se guía un sistema para lograr unos objetivos, expresados como resultados dentro de unos parámetros definidos previamente. Para abordar el problema

de manera ordenada, se debe considerar algunos aspectos básicos sobre el proceso. Es importante tener en cuenta que el control puede tener un objetivo a corto espacio y tiempo, como por ejemplo lanzar una alarma en cado de una intrusión en un hogar, en el caso de un sistema de control domótico, o evitar un obstáculo inmediato un robot en el caso del control de navegación de un robot móvil. En tal caso es habitual hablar de control reactivo. A medida que el ámbito espaciotemporal del control del sistema se va ampliando, se va tendiendo al control deliberativo. Actualmente, debido a los avances en la investigación de sistemas distribuidos basados en la colaboración, se habla también de control colaborativo. En el control colaborativo, los objetivos son compartidos y organizados entre varios componentes del sistema. El ofrecimiento de las funcionalidades que puede tener un objeto se suele realizar por medio de servicios, por lo que las arquitecturas deben incorporar éstos aspectos a su funcionalidad. En estos casos se habla de arquitecturas orientadas a servicios o SOA (Service Oriented Architecture).

Cabe destacar que los componentes de la arquitectura, incluyendo la información que se requiere cambian ampliamente desde el ámbito reactivo al ámbito deliberativo. En el ámbito reactivo la información que se requiere es, sobre todo, información local, proporcionada por sus propios sensores, tanto internos como externos. A medida que los requerimientos del ámbito del sistema a controlar aumentan, los requerimientos de información que se precisan son mayores, tanto en cantidad como en calidad de la misma. Por ejemplo, para detectar una intrusión en un hogar, el sistema domótico sólo debe reaccionar a la activación de uno, o pocos más, sensores de presencia; sin embargo, cuando se trata de realizar una gestión para optimizar el consumo energético del hogar, se debe tener en cuenta los valores de muchos sensores y de muchas mediciones a lo largo del tiempo, por lo que el volumen de información, y la calidad de la misma se hace necesaria.

Para el realizar un control inteligente de un sistema, es necesario poder efectuar muchas acciones, por ejemplo para el control energético de un sistema domótico será necesario tomar ciertas mediciones del consumo, realizar cierta predicción a partir de la comparación con ciertos patrones, y tomar decisiones acerca de qué componentes del sistema tienen prioridad para consumir energía, y cuales deben reducir su consumo. En el caso de la navegación de un robot móvil, se deberán realizar representaciones del entorno para localizar o planificar el robot y hacer que los componentes del robot deban estar preparados para ello, tanto la parte hardware de sensores y actuadores, como el software que maneje el hardware. Esto hace que la arquitectura del sistema que se implemente, será la que determine la capacidad de navegación del robot [Orebäck and Christensen, 2003], siendo esta la base que le proporciona funcionalidad al sistema [Coste-Manière, 2000].

2.2 Escenarios de aplicación

Para determinar qué escenarios conviene implementar, se ha organizado los posibles escenarios en función de los requisitos temporales de la información (desde el tiempo real estricto hasta el tiempo real no estricto) y de los requisitos espaciales de la información (desde un bajo volumen de datos hasta un alto volumen de los mismos). Esta organización se puede ver resumida en la figura 2, y da lugar a diversos escenarios que básicamente se pueden organizar en cuatro áreas. Todos estos sistemas son distribuidos, con la complejidad que ello implica.

En el nivel más sencillo se tienen aplicaciones domóticas, donde las necesidades de tiempo son bajas, ya que la reacción ante un cambio de las entradas suele no tener plazos definidos con una precisión menos al segundo. El volumen de información es escaso, ya que el número de sensores de los que dispone el sistema es escaso y los actuadotes suelen ser menos numerosos. Si se comienza a requerir restricciones temporales más estrictas se puede usar como ejemplo los sistemas deliberativos de navegación de robots. El aumento de los requerimientos temporales sitúa al sistema en el escenario de la navegación reactiva de robots.

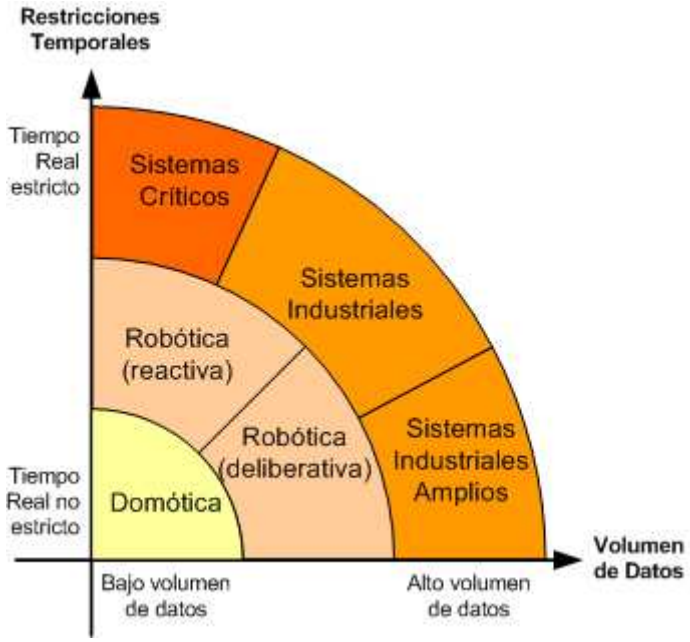

Figura 2. Características temporales y de información en los sistemas de control distribuido.

Un paso más en la complejidad se da cuando el volumen de datos que requiere el sistema va aumentando progresivamente. En este caso, se pasa al ámbito de los sistemas complejos, entre ellos cuando el volumen es alto, pero los requisitos temporales no son estrictos se entra en los sistemas de minería de datos donde es más valioso el resultado de procesar un gran volumen de información que la velocidad con la que se obtiene éste. A continuación, a medida que la velocidad de respuesta pasa a ser tan importante como la eficiencia en el resultado, se entra en el terreno de los sistemas industriales.

Los sistemas con unos requisitos temporales medios pueden ser terminales de contenedores y similares, donde los algoritmos de ubicación de contenedores o de los medios que los transportan son complejos, con costes temporales altos, pero cuyo resultado óptimo justifica la no obtención de resultados en plazos cortos de tiempo. Finalmente las plantas industriales complejas, como las petroquímicas o las nucleares serían el escenario más complejo, donde el volumen de datos a manejar es enorme, y las restricciones temporales críticas.

3 Arquitecturas de control domótico

3.1 Domótica

3.1.1 Domótica y control inteligente

La domótica proviene de la unión de las palabras domus (casa en latín) y robótica (robota, esclavo en checo) aunque algunas fuentes hablan de automática en lugar de robótica. Actualmente se entiende por domótica al conjunto de sistemas que automatizan un hogar, aunque poco a poco se está extendiendo a la automatización de un hogar a un entorno habitable, como oficinas o edificios. En éste último caso, en lugar de hablar de "domótica", se habla de "inmótica". Estos sistemas suelen componerse a partir de sensores comunicados por redes interiores y exteriores de comunicación [Petriu et al., 2000]. En cuanto a las referencias en investigación, el convenio actual es llamar "home automation" a la automatización de un entorno habitable, y "smart home" cuando a dicha automatización se le añade un control inteligente de un hogar.

La línea de investigación más actual es la conocida como "entornos inteligentes" o "smart environment", término que es más genérico que los anteriores. Con éste último término se pueden englobar muchos más sistemas de control inteligente, lo que le da un ámbito de aplicación mayor.

El control inteligente de hogares o entornos es, en la mayor parte de las ocasiones, un control reactivo básico, como puede ser la activación de alarmas, o el encendido y apagado de luces en función de la detección de presencia. Sin embargo, dada la particular configuración de sensores distribuidos de los sistemas domóticos, parecen ser unos escenarios idóneos para la aplicación de nuevas tendencias de inteligencia ambiental y computación ubicua [Ruyter et al., 2005]. Poco a poco se están integrando en estos sistemas el uso de computación más avanzada para determinar sistemas que se adapten a los requisitos de seguridad, comodidad o eficiencia energética que defina el usuario.

3.1.2 Componentes

Se define un entorno inteligente como aquel que es capaz de adquirir y aplicar conocimiento del entorno y sus ocupantes para mejorar sus experiencias en dicho entorno [Cook and Das, 2007]. Los componentes típicos de un sistema de estas características se pueden ver en la figura 3. La automatización de un sistema domótico puede verse como ciclo de percepción del estado del entorno, razonamiento acerca del estado y actuación sobre el entorno. La percepción y actuación del entorno es responsabilidad de la capa física. Estos dispositivos precisan de una interfaz hardware que maneja el sistema operativo correspondiente. Para comunicar los datos de los sensores o las órdenes que se dan a los actuadores, se precisa de una infraestructura de comunicaciones y un sistema operativo que las soporte.

Para comunicar los datos de los sensores o las órdenes que se dan a los actuadores, se precisa de una infraestructura de comunicaciones y un sistema operativo que las soporte. Esta parte es crítica, pues las comunicaciones deben ocultar los detalles de la distribución de los componentes y de los algoritmos.

Aspectos como la ubicación real de un componente, o el enrutamiento correcto de los mensajes deberán ser contemplados. Para ello, los sistemas de comunicaciones suelen basarse en servicios que ofrecen a los componentes. La eficiencia en los servicios que las comunicaciones proveen, da lugar a la eficiencia en los servicios que el control del sistema proporciona a los usuarios. En éste aspecto es especialmente importante el concepto de calidad de servicio.

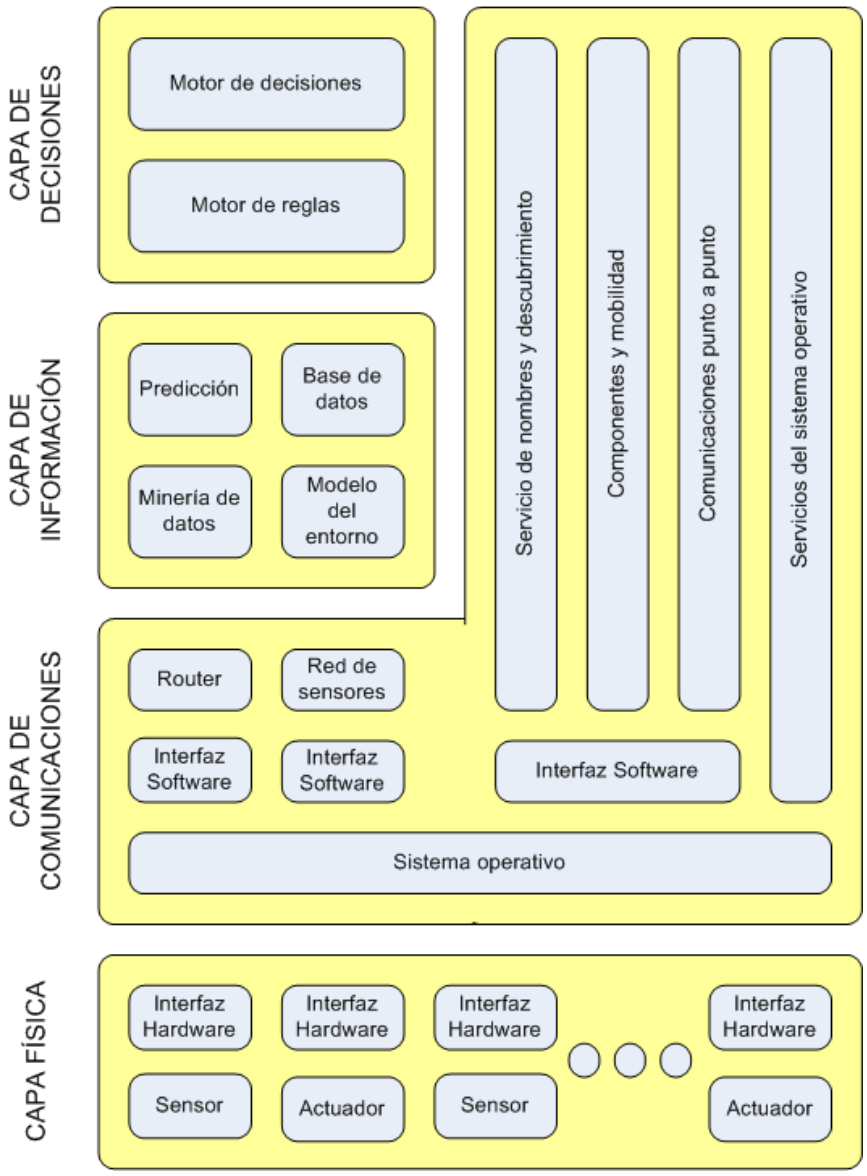

Figura 3. Componentes de las arquitecturas domóticas [Cook and Das, 2007].

Las capas de información y la de decisiones se corresponden con los conceptos clásicos de datos y algoritmos. En estas dos capas, habitualmente considerada una sola en sistemas no distribuidos, se ubica la parte inteligente del control. Habitualmente para la toma de decisiones se recurre a los servicios de la capa de comunicaciones para recibir o enviar información. En lo que respecta a la información es importante destacar tanto la importancia de los datos que se transmiten como de los meta datos, es decir la información acerca de la información. Esta cuestión desarrolla aspectos que van más allá del contenido y se centran en la importancia de los aspectos que rodean a los datos, como los temporales o la calidad de la información.

3.2 Características y evolución

La heterogeneidad de los sistemas de control inteligente, y especialmente en los sistemas domóticos hace que sea complicado determinar una serie de características comunes de los sistemas. En [Aiello, 2005] se destacan las siguientes características que se deben evaluar en un sistema domótico para poder clasificarlo y determinar las capacidades potenciales del mismo.

- Apertura. Se entiende en domótica como la publicidad en el sistema que se da del protocolo, y la posibilidad de implementarlo en cualquier localización.
- Escalabilidad. Se habla de escalabilidad de un sistema domótico como la posibilidad de agregar o remover dispositivos a una red doméstica sin afectar a sus funcionalidades y su rendimiento. Se habla de escalabilidad dinámica en el caso de que se puedan agregar o quitar dispositivos cuando el sistema está en funcionamiento normal.
- Heterogeneidad. Un sistema domótico es heterogéneo cuando la infraestructura del sistema soporta diferentes dispositivos hardware, redes, sistemas operativos, lenguajes de programación del control.
- Topología. La topología de un sistema domótico es la forma en que los dispositivos están conectados unos con otros (topología física, como punto a punto, bus y similares) y cómo se relacionan unos con otros (topología lógica, como cliente-servidor, igual a igual).

Basándose en el grado de apertura, escalabilidad, heterogeneidad y el tipo de topología, se puede clasificar el tipo de sistema domótico. Además, históricamente algunas combinaciones se han dado más que otras. Es difícil clasificar un sistema domótico aunque el punto de vista de evolución temporal puede parecer adecuado para tener una serie de escenarios desde los que es posible ubicar o encuadrar más correctamente una arquitectura concreta. En la figura 4, extraída de [Cook and Sajal, 2007], se puede observar esta evolución y los escenarios que poco a poco van creándose.

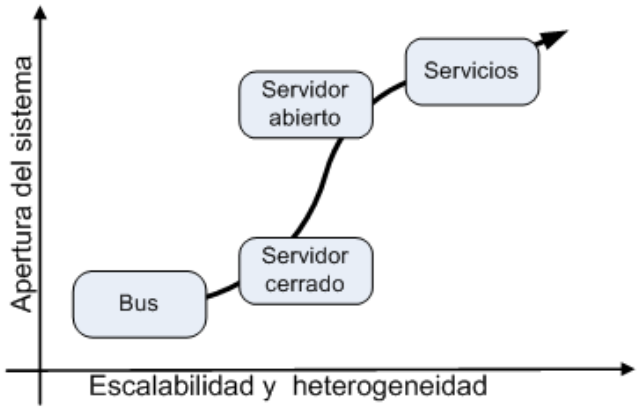

Figura 4. Evolución de las comunicaciones en domótica [Cook and Sajal, 2007],

Inicialmente los sistemas eran poco abiertos y sin posibilidades de ser fácilmente escalables, generalmente basados en buses, en que la escalabilidad del sistema era la propia del bus, y la posibilidad de crear un sistema heterogéneo mínima, ya que solo los dispositivos compatibles con el bus tienen posibilidad de conectarse.

Poco a poco se crean sistemas donde el control se centraliza para proporcionar un control externo. A continuación aparecen sistemas basados en servidores que proveen servicios públicos independientemente del sistema que controlan, de esta manera el sistema pasa a tener un nivel de apertura elevado. Actualmente se está trabajando en lograr una mayor escalabilidad del sistema a partir de arquitecturas basadas en servicios, especialmente en los servicios Web.

3.2.1 Sistemas basados en bus único

Los sistemas domóticos centralizados y los basados en un bus son los que se conectan todos los dispositivos a un mismo medio, tal como se ve en la figura 5. Es habitual que el control lo realice un solo dispositivo conectado al bus. El protocolo que se emplea es cerrado, con escasa extensibilidad hacia otros protocolos y la escalabilidad está limitada a un número concreto de dispositivos. La heterogeneidad es escasa ya que la limitación del tipo de bus que se impone hace que los dispositivos deban estar especialmente diseñados para ese bus.

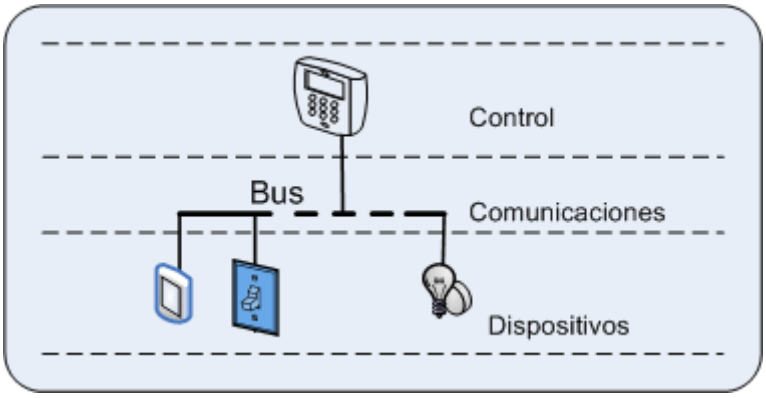

Figura 5. Sistema domótico basado en un bus único.

El ejemplo más habitual de este tipo de escenario son los sistemas que emplean el protocolo X-10 donde el bus es la línea de tensión, la velocidad y el número de dispositivos están limitados. El modelo ha evolucionado hacia sistemas basados en EIB o la tendencia actual de Konnex. Esta evolución ha ido incrementando la potencia del bus y la escalabilidad, al permitir más dispositivos y más protocolos.

3.2.2 Sistemas basados en servidor

La necesidad progresiva de tener mejores accesos a los diferentes dispositivos de un hogar y el hecho de que haya variaciones de protocolos y características entre distintos fabricantes, hace que el modelo basado en un solo bus no sea el más adecuado. Para poder acceder a los dispositivos con distintas necesidades tecnológicas, se planteó la posibilidad de tener unos dispositivos exclusivos cuya misión es dar soporte a las diversas tecnologías que se pueden dar en un mismo sistema, estos dispositivos son los servidores. Puede verse un ejemplo de éste tipo de sistema en la figura 6.

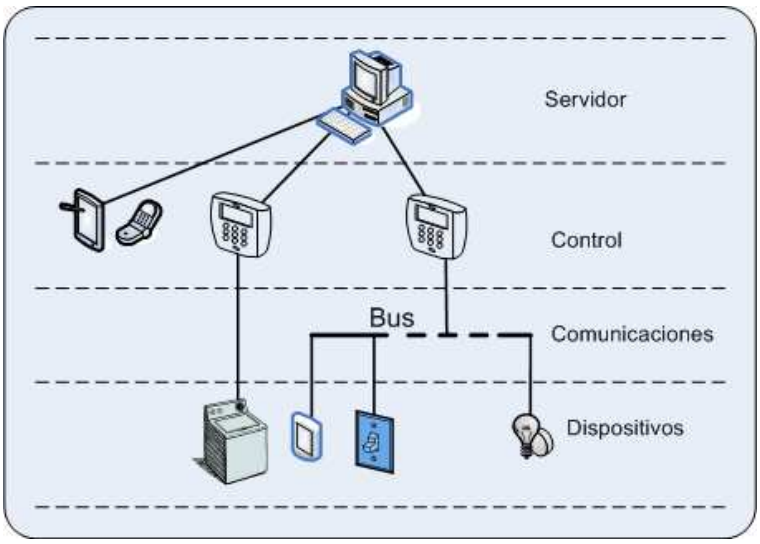

Figura 6. Sistema domótico basado en un servidor.

Lo más habitual es que el modelo se base en un solo servidor que controla los dispositivos del sistema, en tal caso se conoce como sistema de servidor centralizado. Algunos sistemas como Lonworks, emplean un sistema centralizado con una topología cliente-servidor.

La evolución del modelo ha dado lugar a emplear servidores que soporten diversas tecnologías y que, de esta forma, se puede monitorizar y gestionar los dispositivos, aunque sean heterogéneos, desde distintos lugares. En caso de tener los servidores en diversos elementos del sistema se habla de un sistema de servidor distribuido.

3.2.3 Sistemas basados en servicios

La necesidad de comunicar a distintos niveles los sistemas, hace que la comunicación no deba ser exclusivamente cliente-servidor. La evolución del sistema cliente-servidor ha dado lugar a los sistemas basados en servicios, lo que no excluye el empleo de buses, ni de controladores cliente-servidor. En el modelo basado en servicios los dispositivos se ofrecen funcionalidades entre ellos (figura 7) de forma que un dispositivo ofrece unas funciones y puede emplear las funciones de otros. Se puede decir que en estos sistemas, cualquiera de los elementos puede hacer de servidor y de cliente. Este modelo se conoce como de igual a igual o “peer to peer” donde los componentes se ofrecen servicios entre ellos.

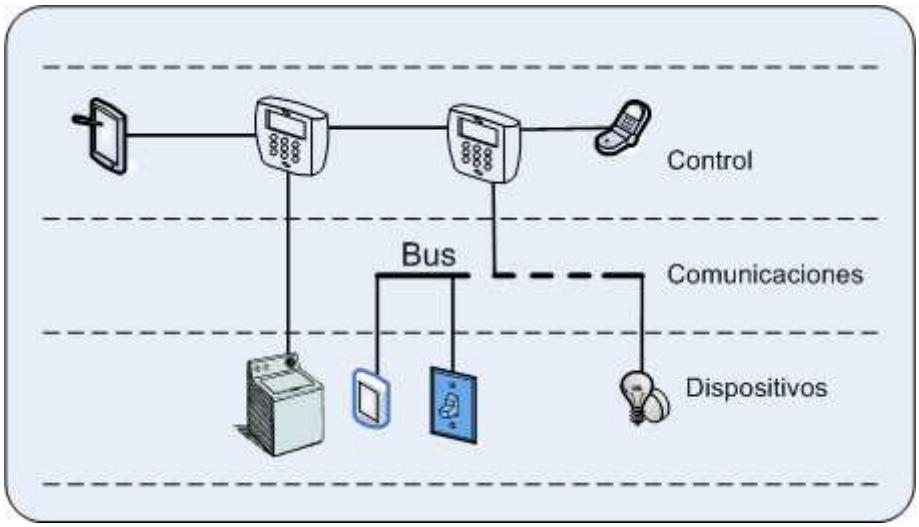

Figura 7. Sistema domótico basado en servicios.

Actualmente se puede localizar más fácilmente las tecnologías que proporcionan a los dispositivos la posibilidad de ofrecer servicios (Jini, etc) que arquitecturas domóticas con posibilidad de estandarizarse.

3.3 Revisión de arquitecturas de control domótico

La revisión se ha planteado desde un punto de vista de arquitecturas que se aplican en sistemas domóticos. Se debe tener en cuenta que actualmente los nombres de la tecnología se varían según autores, entre los nombres están: Home automation, Smart home, Domotic, Home control, Smart house, Intelligent home, House control, Intelligent house, House Automation o Smart environment. Parece ser que el convenio es llamar "home automation" a la automatización de un entorno habitable, y "smart home" cuando a dicha automatización se le añade un control inteligente de un hogar. Es de destacar que, últimamente está progresando el uso del término "smart environment", término que es más genérico que los anteriores y se refiere a entornos inteligentes. Con éste último término se pueden englobar muchos más sistemas de control inteligente, lo que le da un ámbito de aplicación mayor.

Debido a que las arquitecturas de control domótico suelen clasificarse, bien por la ubicación del control (centralizado o distribuido) o por el tipo de medio por el que se comunican los componentes (bus, servidor o servicio), se puede hacer una revisión por cualquiera de ellas. Las siguientes arquitecturas se describen en orden cronológico desde la primera referencia bibliográfica de cada una de ellas de esta manera se puede observar la evolución de las mismas que se describió en la figura 4.

3.3.1 ACHE

ACHE (Adaptive Control of Home Environment) es la arquitectura planteada en [Mozer, 1998] y analizada en [Mozer, 2004]. El objetivo de la arquitectura ACHE es lograr una automatización y predicción del comportamiento de los habitantes de la casa. Para ello comprueba los ajustes manuales que realizan los ocupantes y emplea los valores como referencias para el entrenamiento para el aprendizaje de sus costumbres. Además del objetivo de aprendizaje tiene como segundo objetivo el ahorro energético, variando la intensidad de las luces, controlando la presencia en habitaciones, temperaturas y métodos similares.

Para lograr la eficiencia de los objetivos, se basa en un concepto llamado “entorno de control óptimo” en el que pretende lograr los objetivos con un coste mínimo. Para ello se define un coste de “des-confort”, que se da cuando un ocupante configura los parámetros del entorno fuera de los parámetros de eficiencia que ha calculado ACHE. Además también tiene en cuenta un coste energético que se basa en el consumo de las fuentes energéticas por parte de los usuarios. Para la implementación de ACHE se emplea una arquitectura que se replica para cada “dominio de control”: iluminación, control de temperatura, etc. El esquema con los componentes puede verse en la figura 8.

Para conocer el estado actual del entorno, se emplea una transformación del estado, consistente en el cálculo de parámetros como promedios, valores máximos y mínimos y similares. El resultado es una representación del entorno que describe en mejor medida que el estado de los valores instantáneos. Los valores instantáneos también se envían a un “modelo de ocupación” que determina la ocupación de las zonas del entorno a controlar. A continuación se encuentran tres componentes adaptativos. El primero de ellos es el “predictor” que a partir del estado actual busca posibles estados deseados. Ejemplos de predicciones son, las expectativas de movimientos de ocupantes del entorno, el posible uso de energía o agua y cuestiones similares. Estas predicciones se realizan mediante redes neuronales.

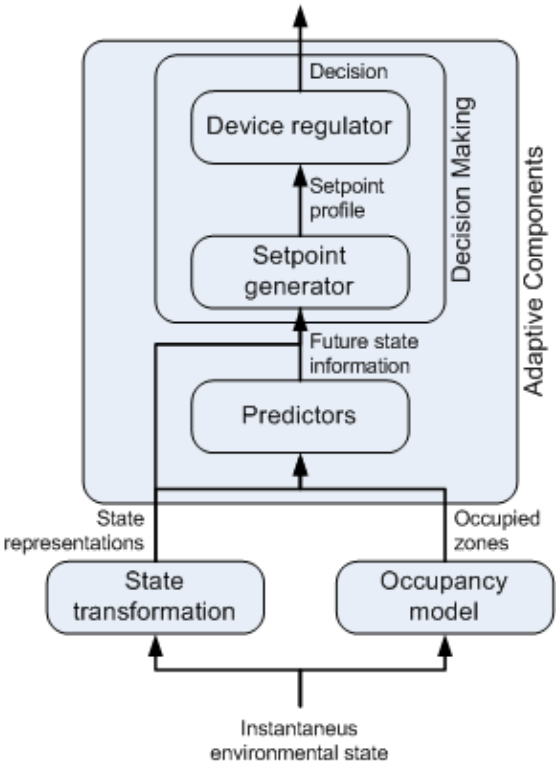

Figura 8. Arquitectura domótica ACHE.

La toma de decisiones acerca de cómo configurar los actuadores, se toman por medio de dos componentes con el objetivo de encapsular el proceso de generación de conocimiento. El primero de los componentes es el “setpoint generator” que determina el objetivo de una de las variables de control (temperatura, nivel de iluminación, etc.) sobre una ventana temporal en la que se desea que dicho valor se alcance. El segundo componente es el regulador de dispositivo, que controla directamente los dispositivos a partir de los valores prefijados.

3.3.2 Home API

Home API [Bizarri, 1999] es el resultado de un grupo de trabajo fundado en 1997 con 31 participantes, entre los que destacan Compaq, Intel, Honeywell, Mitsubishi Electric, Phillips y Microsoft. El lanzamiento oficial fue en octubre de 1998. La intención del grupo es establecer una especificación abierta que permita monitorizar y gestionar dispositivos domóticos entre el cliente y el sistema, tal como se ve en la figura 9. En un principio definen "Home API" como:

- Servicio que funciona bajo MS Windows.
    - Permite el descubrimiento y el control de dispositivos del hogar por medio de aplicaciones Windows.
    - Aislado del protocolo de red empleado.
- Entorno para gestionar dispositivos por medio de espacios de nombres.
- Aplicaciones que instalan comportamientos en el hogar.

En las especificaciones aclaran que no se trata de un servicio de gestión de red y que no es un sistema implicado directamente en las transmisiones multimedia. Como arquitectura, Home API, está diseñada para el control de dispositivos domóticos, sin emplear nuevos protocolos ni añadir nuevas redes a las ya existentes, aunque en los ejemplos emplea IEEE 1394 (Firewire).

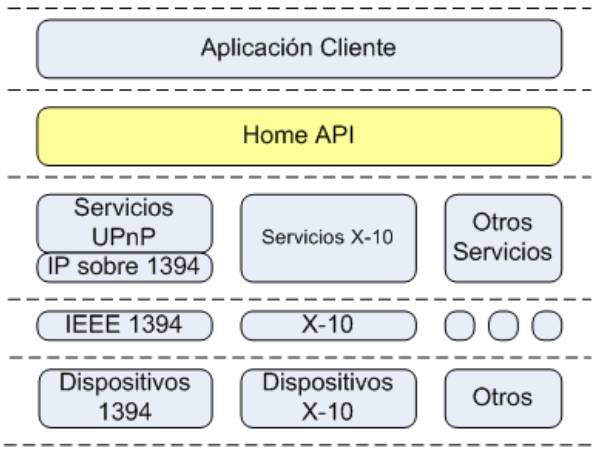

Figura 9. Ubicación de la arquitectura domótica Home API.

Como servicios, Home API debe proveer los siguientes:

- Creación de objetos por medio del descubrimiento y con capacidad de control Para ello se emplea la tecnología COM/OLE.
- Rutas de propiedades, por medio de la propagación de cambios de estados.
- Eventos y subscripciones. Con obtención de los datos por actualización y bajo demanda.
- Contendores, que encapsulen el contexto y el comportamiento.
- Asociaciones entre los componentes relacionados.
- Operaciones asíncronas, con tolerancia a fallos.

El uso de un entorno de espacio de nombres se emplea para reflejar las topologías y las topografías del hogar. Por medio de este espacio de nombres se localiza un dispositivo dentro del hogar. Un ejemplo de espacio de nombres puede verse en la figura 10.

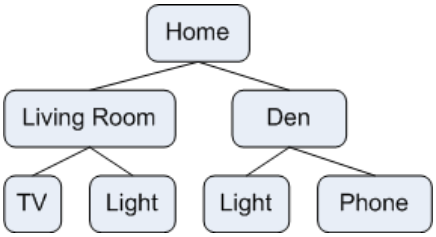

Figura 10. Modelo de espacio de nombres empleado en Home API.

Los comportamientos inteligentes del sistema, se encapsulan en unos contenedores conocidos como "Home API Process" estos procesos usan rutas para describir la relación entre las propiedades de dos objetos, por ejemplo, una activación de interruptor que encendiese una luz, se reflejaría como la siguiente relación:

Ecuación 1. Ejemplo de uso de espacios de nombres en la arquitectura Home API.

$$\text{"mySwitch.Power} - \text{myLight.Brightness"} \quad (1)$$

HomeAPI está pensado para permitir a las aplicaciones que lo utilicen poder controlar o conocer el estado de un dispositivo, aunque tal como se especifica inicialmente no soporta el envío de flujo de datos tales como vídeo. HomeAPI se basa en un modelo de control centralizado en el que un número pequeño de nodos inteligentes controlan numerosos dispositivos. Cada dispositivo se modela mediante una serie de objetos OLE que tienen un conjunto de propiedades, que se pueden consultar o modificar. Un gestor de eventos permite avisar a las aplicaciones (llamadas nodos) subscritas a los eventos, sobre los cambios en una propiedad. Para hacer compatible sus productos, los desarrolladores de hardware sólo deben incluir, junto al dispositivo, el objeto OLE correspondiente que lo controla, de forma que cualquier aplicación pueda acceder fácilmente a través de la interfaz estándar de Microsoft. Los nuevos dispositivos también se pueden modelar a partir de objetos OLE ya definidos sin más que añadir nuevas propiedades.

3.3.3 Aladdin

Aladdin [Wang et al., 2000] es una arquitectura de inter-conectividad (networking) para un sistema domótico. La arquitectura software se divide en tres capas; pueden verse, junto a los componentes que la integran en la figura 11. La capa inferior se denomina capa de infraestructura del sistema que está asociada con un sistema de suscripción y publicación (Pub/Sub eventing) de eventos que conforma el (Soft-State Store). Un servicio de nombres y atributos que mantiene una tabla con todos los dispositivos que se encuentran funcionando. Un protocolo de anuncio de dispositivos y una serie de procesos que gestionan los fallos del sistema.

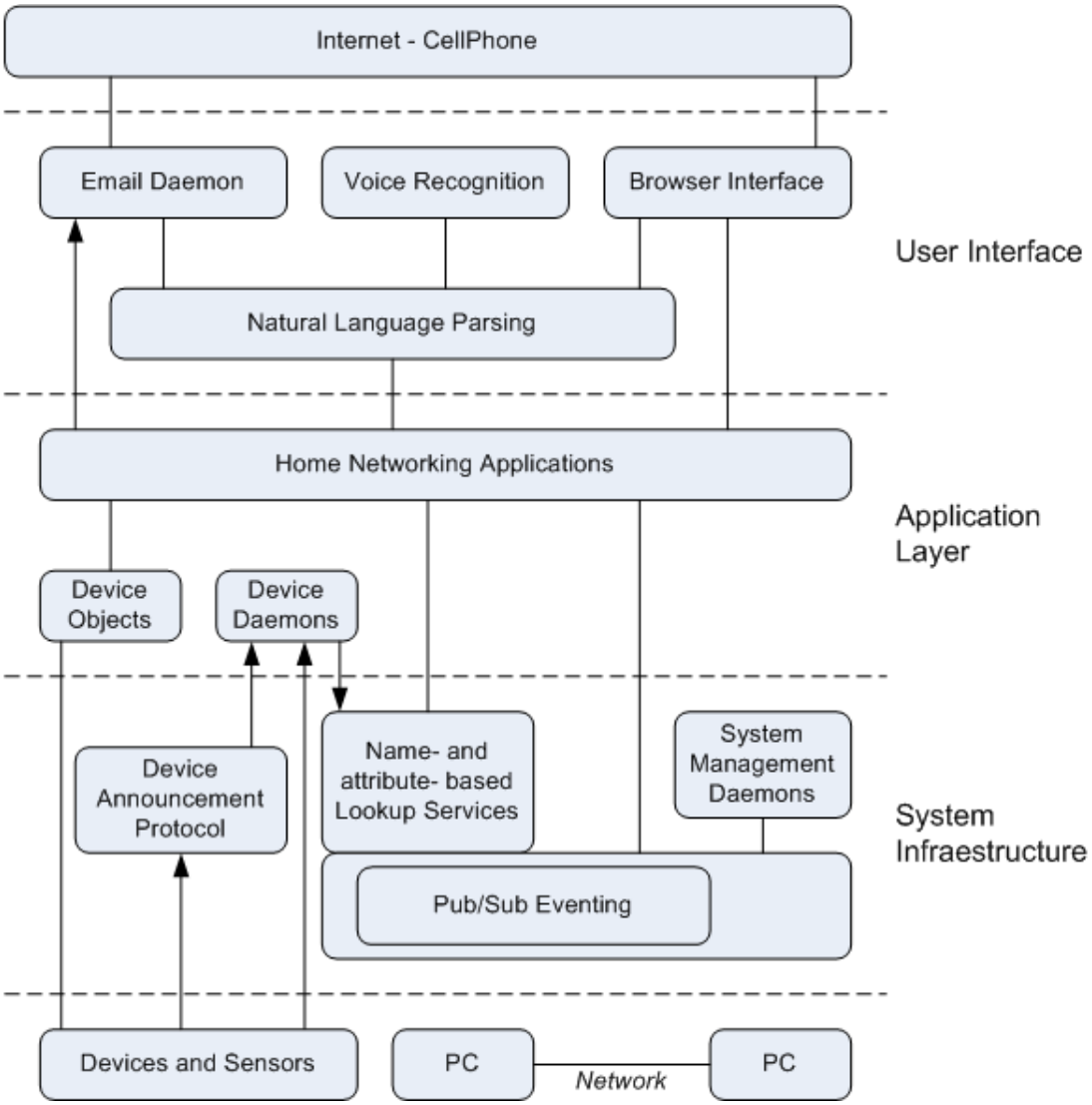

Figura 11. Arquitectura domótica Aladdin.

La capa intermedia es la capa de aplicación, consiste en una serie de objetos que representan los dispositivos y procesos de dispositivos que encapsulan los detalles específicos de los dispositivos y la red a las aplicaciones de control domótico. La última capa, la interfaz de usuario, permite al usuario configurar y controlar el sistema.

3.3.4 MASSIHN

MASSIHN (Mobile Agent Space System for Intelligent Home Network) [Cheng-Fa and Hang-Chang, 2002] es una arquitectura de red que da soporte a la integración de un sistema multiagente con los componentes de un sistema domótico controlados por un computador. Está programado en JAVA y la información entre agentes se intercambia en formato XML.

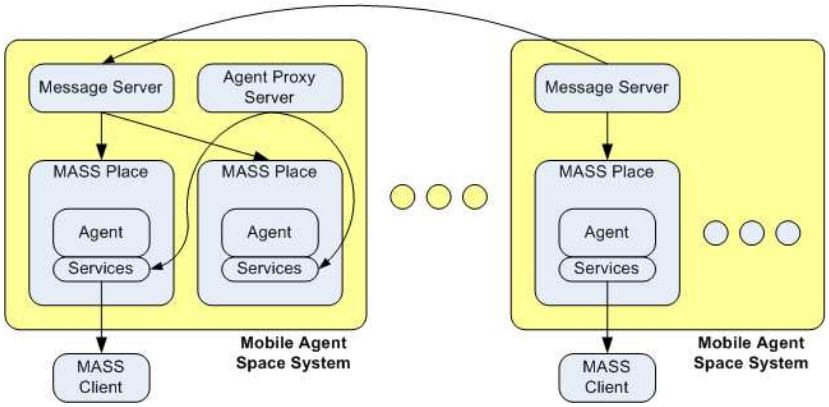

Figura 12. Arquitectura domótica MASSIHN.

La arquitectura tiene seis componentes principales, pueden verse, junto a las relaciones entre ellos en la figura 12. La funcionalidad de los componentes es la siguiente:

- Mesage Server. Es una de las partes importantes en el sistema multiagente, opera como un objeto RMI. Es preciso que cada "MASS Place" se registre en el servidor de mensajes. El "Message Server" proporcionará los servicios de comunicación entre todos los "Message Server".
- Agent Proxy Server. Se encarga de realizar las transacciones para cargar y almacenar varios agentes en el "MASS Place"
- MASS Place. El "MASS Place" es el entorno donde los agentes realizan las principales acciones, soporta diversos servicios para los agentes, una de sus tareas es registrar el entorno en el servidor de mensajes. Además debe traducir mensajes de XML y configurarse en función de las características requeridas.
- Services. Los servicios son las funcionalidades que ofrece el "MASS Place" a los clientes. Los servicios que ofrece son los siguientes.
    - Perform. Este servicio acepta comandos remotos en XML y carga el agente correspondiente para realizarlo. Este servicio no es el encargado de mover agentes.
    - Dispatch. Este servicio es el que proporciona el punto de partida para que los agentes atraviesen la red. Este servicio recoge las características de los diferentes puntos para que se pueda tomar decisiones acerca de los desplazamientos de agentes.
    - Dispatch and perform. Este servicio es el que envía los comandos al resto de agentes, y además decide la asignación de éstas.
- Agents. El sistema proporciona tres tipos de agentes.
    - Service UI. Este agente es el que proporciona la interfaz a los usuarios finales.
    - Perform Agent. Es el agente encargado de cargar y recoger el contenido de las tareas que los usuarios han asignado.
    - Service Agent. Representa el centro de operaciones del servicio.
- MASS Client. Este componente obtiene la información del "Message Server" y prepara un contenedor para cargar la GUI del servicio

Además de los componentes anteriores, en la arquitectura se presenta una configuración interesante con la incorporación de un generador de eventos que permite probar el sistema en condiciones diferentes a las habituales. El funcionamiento puede observarse en la figura 13.

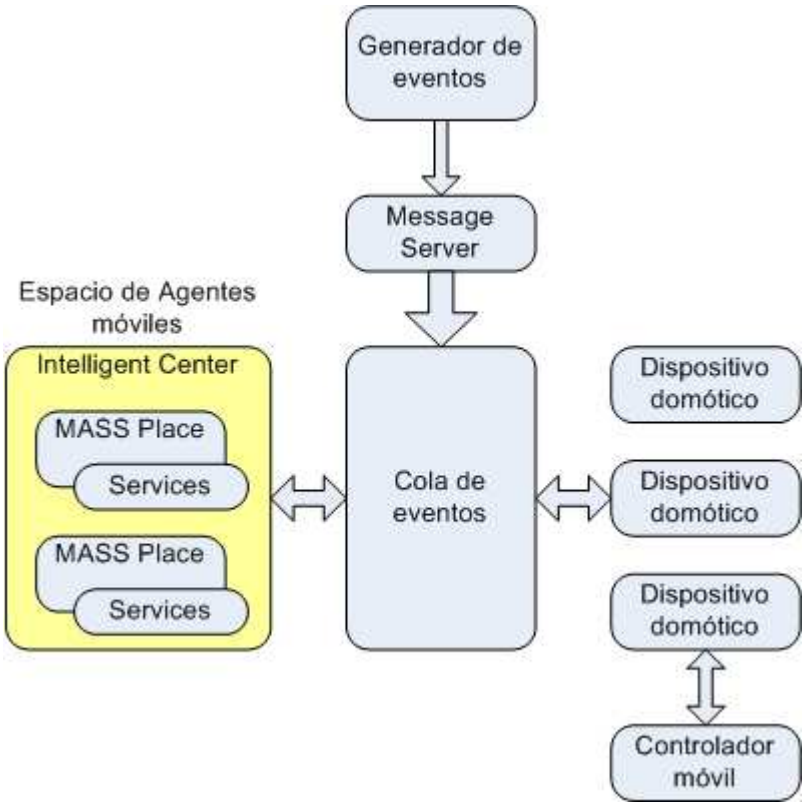

Figura 13. Generador de eventos de la arquitectura MASSHIN.

El generador de eventos se conecta a un servidor de mensajes para ocultar el hecho de que estos mensajes procedan de un sistema simulado. Estos mensajes pasan a una cola de eventos que es la responsables de comunicar el sistema real domótico con el sistema multiagente que lo gestionará.

3.3.5 MavHome

MavHome (Managing An intelligent Versatile Home) es un proyecto [Cook et al., 2003] multidisciplinar de la Universidad de Texas. Con una arquitectura muy sencilla y un interesante método de localización en el entorno [Abhishek et al., 2003].

MavHome se define como un sistema que representa un entorno que actúa de manera inteligente como un solo agente. El objetivo del sistema es maximizar el confort de sus ocupantes minimizando los costes de operación. Para ello, el entorno debe ser capaz de predecir, razonar y adaptarse a los requerimientos de sus ocupantes. Par apoder gestionar la gran cantidad de áreas que representan un entorno inteligente, MavHome divide el agente de control en agentes de más bajo nivel. La relación interna de cada componente es por capas, mientras que entre componentes se mantiene una jerarquía. Puede verse un ejemplo en la figura 14.

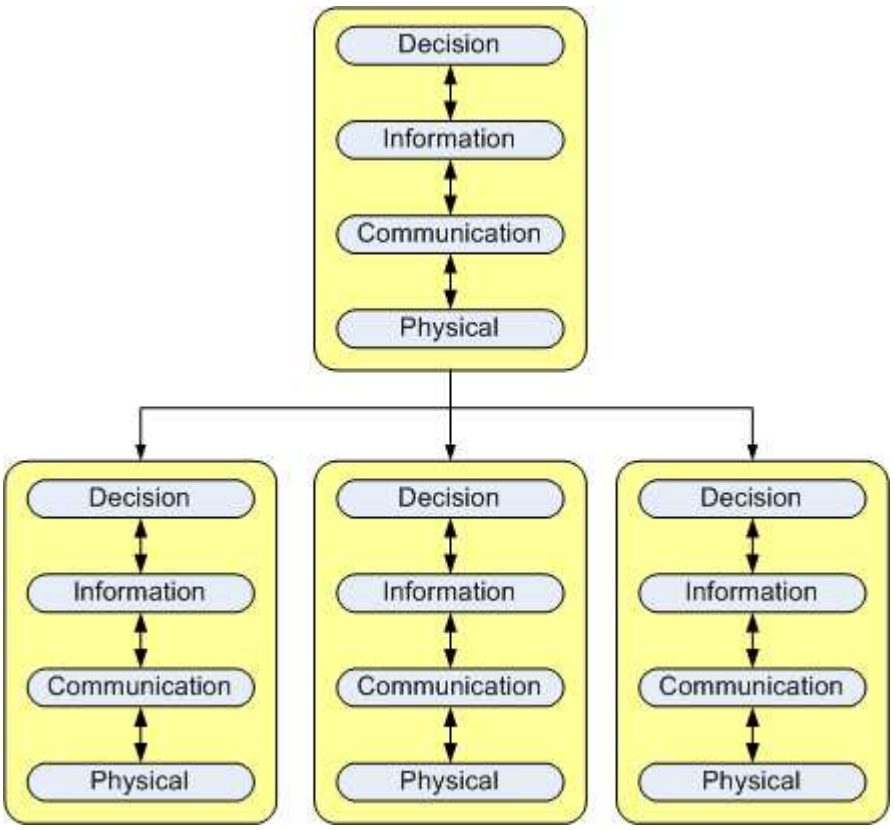

Figura 14. Arquitectura domítica MavHome.

El sistema de cada agente se divide en cuatro capas con funciones muy específicas para cada una de ellas. La capa superior es la capa de decisiones, y selecciona las acciones que llevará a cabo el agente a partir de la información que le ha sido proporcionada por las capas inferiores a través de la capa de información. La siguiente capa es la capa de información. Esta capa se encarga de generar el conocimiento relevante para la toma de decisiones del sistema. Debajo de la capa de información se encuentra la capa de comunicación, esta capa facilita la integración de los datos que deben transferirse entre los agentes para poder ser relevantes a la capa de información. Finalmente se encuentra la capa física, que contiene el hardware de sensores y actuadores y de comunicaciones entre ellos y con las capas superiores.

3.3.6 C@sa

C@sa [De Carolis and Cozzolongo, 2004] es un sistema multiagente cuyo objetivo es modelar, controlar y simular el comportamiento de un sistema domótico. Para lograr el control, c@sa divide el control en diversos grupos de agentes correspondientes con las diversas áreas de acción. Los componentes y su distribución pueden observarse en la figura 15.

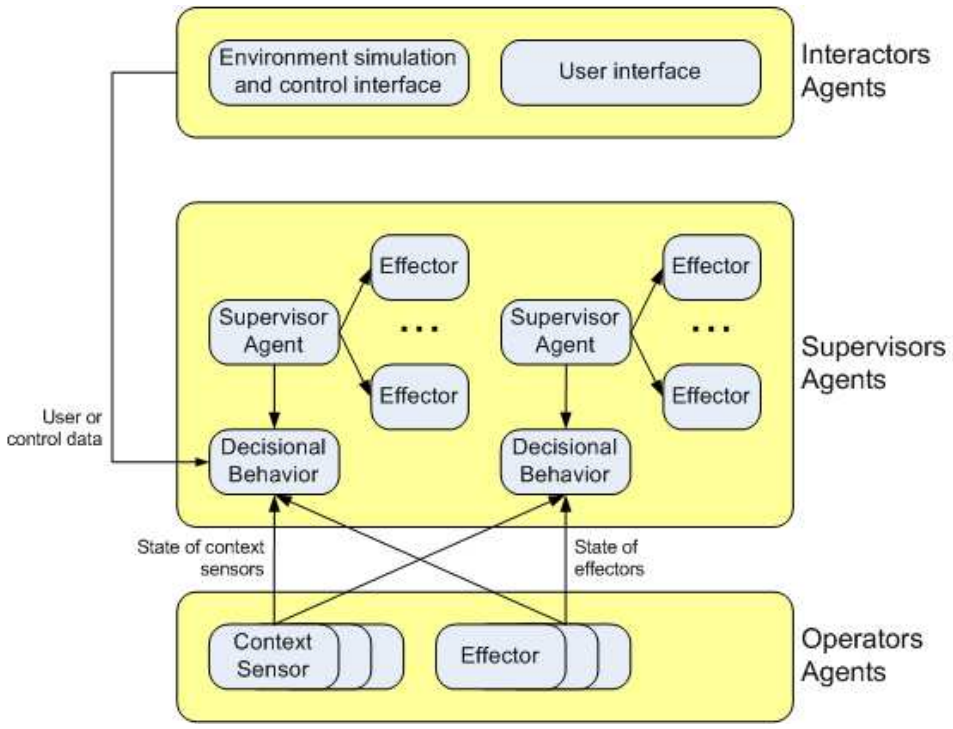

Figura 15. Arquitectura domítica c@sa.

Los agentes más cercanos al sistema son los "operador agents", que deben controlar y modelar el comportamiento de un dispositivo simple. Se define como un conjunto de atributos que describen las características del mismo y una serie de comportamientos, que describen las tareas que el agente puede realizar a solicitud de un usuario o de otro agente. Los agentes de operaciones pueden ser de dos tipos: sensores de contexto, que miden el valor de una o más características de los sensores y actuadores, que pueden realizar acciones sobre los dispositivos.

Sobre la capa de agentes de operaciones, se encuentran los agentes supervisores, la misión de estos agentes es relacionar diversos agentes operacionales. Estos agentes leen los estados de los sensores y a partir de los parámetros de configuración que tengan deciden las acciones a tomar.

Finalmente se tiene la capa de agentes de interacción. Estos agentes son los encargados de configurar el control del sistema. Pueden ser de dos tipos los de simulación y control y los de usuario. Los primeros son los que se emplean para simular el sistema y conocer el comportamiento del mismo dada una configuración. Los segundos se emplean por el usuario directamente para interaccionar con el sistema.

3.3.7 HAS

HAS (Home Automation System) está descrita en [Chao-Lin et al., 2004] está basada en un sistema de agentes móviles (y sus correspondientes host) que trata de integrar diferentes tecnologías de red para la automatización de un sistema domótico. Realmente la arquitectura está basada en el esquema clásico de agentes móviles (figura 16). El sistema tiene una serie de bases de datos de agentes que se van moviendo a los distintos host. Estos host son los que lanzan los agentes y los mantienen, y pueden ser desde una PDA hasta un ordenador complejo. La información de los dispositivos es transmitida a través de la red correspondiente a una pasarela de protocolo que es la encargada de transformar los mensajes en un formato común a los agentes.

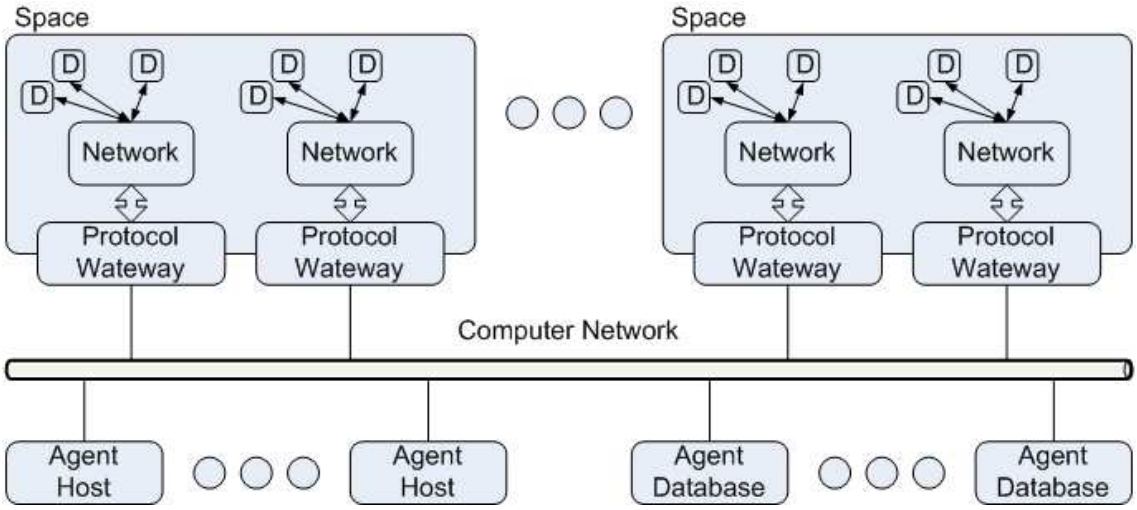

Figura 16. Arquitectura domótica HAS.

El sistema puede contener diversos tipos de agentes, entre los cuales cabe destacar los siguientes.

- Interface Agent. Es el agente responsable de la interacción entre el sistema y el usuario y por tanto debe comunicar al resto de agentes las tareas que los usuarios solicitan al sistema.
- Device Control Agent. Este tipo de agente es el responsable de la comunicación directa con los dispositivos. Su misión es controlar el estado del dispositivo y cooperar con el resto de los agentes respondiendo a los comandos que se le soliciten. Existen una gran cantidad de agentes de control de dispositivo debido a la heterogeneidad de los medios de comunicación empleados.
- Message Agent. Cuando un agente desea comunicarse con algún otro agente remoto, se precisa de un agente de mensajes para gestionar dicha comunicación. La misión de este agente es transportar el mensaje de un agente al correspondiente host.
- Function Agent. Este es el agente que gestiona la labor de control de alto nivel. Son los que deben economizar la energía, satisfacer las necesidades de confort del usuario y labores similares.

La implementación de los componentes de la arquitectura se hace a partir de estos agentes o de combinaciones de ellos. Por ejemplo el "protocol gateway" es un "Agent Host" que contiene variso "Device Control Agent". El "Agent Database" es un "Agent Host" con conexión a un Sistema Gestor de Base de Datos" con la información necesaria para poner en marcha un agente. La comunicación entre agentes se realiza por medio del lenguaje KQML.

3.3.8 DomoNet

DomoNet es una arquitectura de red orientada a evitar los problemas que surgen al emplear los diversos medios de comunicación que se encuentran en los sistemas domóticos [Miori et al., 2006]. Por tanto DomoNet no es tanto una arquitectura de control sino una arquitectura que da el soporte al control. Este control se realiza por medio de servicios Web. El ámbito de la arquitectura puede verse en la figura 17.

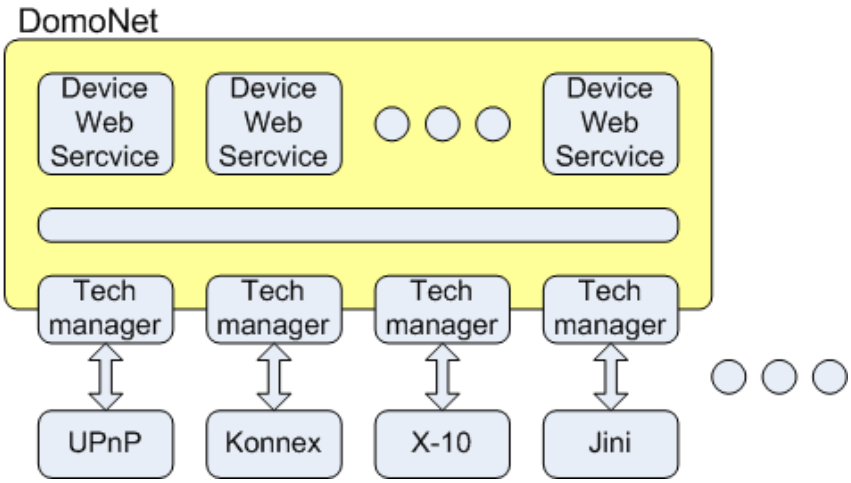

Figura 17. Arquitectura domótica DomoNet.

DomoNet emplea el concepto de "Tech Manager" (figura 18) para gestionar la tecnología de red correspondiente, y se encarga de ofrecer el mismo interfaz a los servicios de red que tiene por encima de ellos, de esta manera hace transparente la tecnología empleada y permite que el control sea realizado por las aplicaciones Web correspondientes. Hay un servicio Web por cada acción de control que se desee realizar, por ejemplo para el control de las luces, para el control de las alarmas de seguridad, etc. De esta manera los dispositivos son gestionados por medio de servicios Web de manera independiente a la tecnología que estos dispositivos emplean.

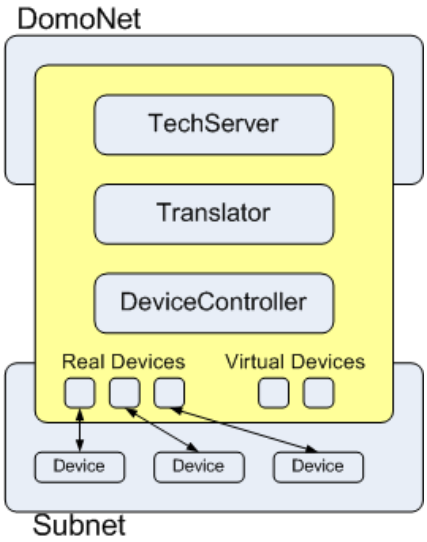

Figura 18. Conexión de DomoNet al sistema domótica.

La estructura interna de un Tech Manager son aplicaciones compuesta por una serie de componentes encargados de interconectar la red domótica (subnet) con la red "DomoNet" que hablará directamente con los servicios Web que controlarán el sistema. Se componen de un "TechServer", que es el componente encargado de gestionar las comunicaciones TCP con los servicios Web de control. Además existe un "device controller" que es el componente que se encarga de controlar el dispositivo por medio del protocolo que este emplee. La traducción entre uno y otro componente la realiza el "translator" que será el encargado de transformar los comandos de un protocolo en comandos del otro.

Se pueden crear diversos dispositivos virtuales para que los clientes o los servicios Web puedan realizar los ajustes necesarios y para poder ser probados los algoritmos de control por medio de simulaciones sin que el sistema pueda distinguir si se trata de un dispositivo real o simulado.

Una de las características que más se mencionan en las referencias a DomoNet es el empleo del lenguaje DomoML para intercambio de información y órdenes dentro del sistema domótico. DomoML es un lenguaje en XML que es común a todos los servicios Web que integren la arquitectura DomoNet.

3.4 Análisis

Comparar arquitecturas de sistemas de control inteligente es complicado, si además se trata de un área reciente de investigación como la domótica, puede resultar difícil determinar los parámetros de comparación. Inicialmente, y en base a las arquitecturas expuestas anteriormente la forma de implementación, la ubicación del control inteligente y la configuración de los sistemas son los siguientes.

Tabla 1. Comparación de las características básicas de las arquitecturas anteriores.

Arquitectura	Paradigma	Control	Implementación	Detalles del sistema comunicaciones
ACHE	Bus	Centralizado	Componentes	-
Home API	Servidor	Centralizado	Componentes	Espacio de nombres
Aladdin	Servidor	Centralizado	Objetos	Publicación/Suscripción
MASSIHN	Servicios	Distribuido	Agentes móviles	Publicación/Suscripción
MavHome	Servicios	Centralizado	Multiagente	-
C@sa	Servicios	Centralizado	Multiagente	-
HAS	Servicios	Distribuido	Agentes móviles	-
DomoNet	Servicios	Distribuido	Servicios Web	Publicación/Suscripción

De la tabla anterior se puede deducir que las tendencias en los sistemas domóticos son los basados en servicios así como las arquitecturas que tengan una capacidad de distribución, ya son también las que proporcionan una potencia mayor. En lo que se refiere a la configuración del sistema, los más potentes son aquellos que proporcionan un sistema de publicación y suscripción para comunicar sus componentes. En cuestión de tecnologías y carácteríticas, las arquitecturas evaluadas emplean las que se muestran en la Tabla 2.

Tabla 2. Características de las diferentes arquitecturas

Característica	ACHE	HomeAPI	Aladdin	MASSIHN	MavHome	C@sa	HAS	DomoNet
Anuncio dispositivos			Sí					
Aprendizaje	Sí				Sí			
Automatización	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Comportamiento		Sí						
Descubrimiento		Sí						
Dominios de control	Sí				Sí			
Espacios de nombres		Sí	Sí					
Eventos		Sí	Sí	Sí				
Gestión de fallos			Sí					
Monitorización		Sí						
Predicción	Sí				Sí			
Repr. Entorno	Sí				Sí			
Servicios		Sí						

4 Arquitecturas de control inteligente de robots móviles

4.1 Paradigmas

Aunque cada sistema a controlar suele tener una arquitectura particular, generalmente, se han desarrollado la mayoría en base a dos paradigmas: el paradigma reactivo y el paradigma deliberativo. Una combinación de ambos paradigmas, conocida como paradigma híbrido, es la que actualmente tiene más aceptación. A continuación se verán los detalles de estos paradigmas.

4.1.1 Paradigma reactivo

El control reactivo se basa en el principio de acción – reacción (figura 19), tiene sus inicios en la observación del comportamiento básico de los animales [Brooks, 1986]. El objetivo que se busca en la navegación reactiva, es el no depender de una capa inteligente para lograr un comportamiento que lo parezca [Brooks, 1991b], por lo que el fundamento básico del paradigma reactivo implica una conexión más o menos directa entre los sensores y los actuadores. Esta conexión directa se basa en la dualidad Sentir – Actuar.

A partir de la existencia de múltiples instancias de procesos Sentir – Actuar se componen los denominados patrones de comportamiento. Un patrón de comportamiento recoge la salida de un sensor y genera unas señales en los actuadores. Las acciones que provoca un comportamiento son independientes de las que puedan generar el resto de comportamientos que se puedan estar dando en el mismo sistema. La ejecución concurrente de varios patrones de comportamiento hace que algunos de ellos sean eclipsados por otros. El resultado es un comportamiento, que no es básico, y que se corresponde a una combinación de comportamientos denominado comportamiento emergente.

Figura 19. Paradigma de control reactivo.

Este comportamiento emergente puede parecer el resultado de un razonamiento, más o menos complejo, y que estará en función del número y tipo de comportamientos que el robot pueda tener programados. Se puede llegar a plantear la cuestión de conocer si una acción deliberada que tenga una complejidad muy elevada, puede ser lograda por medio de este paradigma.

La ausencia de representación interna hace que estos sistemas no precisen realizar localización ni planificación, por ello los sistemas reactivos tienen la gran ventaja de no precisar procesamientos complejos ni requieren estructuras de datos complejas que representen el entorno. Al no ser sistemas dependientes de algoritmos, son sistemas en los que se da una capacidad de predicción temporal que permite cumplir restricciones de tiempo real.

Sin embargo, cabe plantearse qué ocurre en el momento en que se incluya un elemento de memoria en el sistema, puesto que en ese momento se puede estar hablando de representaciones implícitas, aunque muy primitivas, del entorno. En este punto es donde puede llegar a estar hablando de la frontera entre el paradigma reactivo y el siguiente paradigma, el deliberativo.

4.1.2 Paradigma deliberativo

El control deliberativo [Nilsson, 1980] basa en el razonamiento las decisiones acerca de las decisiones que se deben de tomar (figura 20). Aunque sus orígenes datan de los años sesenta y del nacimiento de la inteligencia artificial, actualmente la capacidad de cómputo permite disponer de sistemas más complejos.

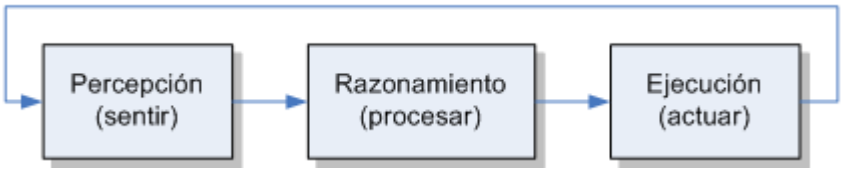

Figura 20. Paradigma de control deliberativo.

Realizar las acciones de razonamiento implica tener una representación interna del entorno y unos algoritmos que, en función de dicha representación y de los datos percibidos, tomen las medidas necesarias para ejecutar las acciones que se hayan decidido. Sin embargo, almacenar toda la información que esté disponible en el entorno y en el sistema es una opción no viable en los sistemas inteligentes [Chown, 1999]. Además, el tiempo que transcurre entre la percepción del entorno y la ejecución de las acciones puede hacer que varíe el entorno y los datos percibidos no sean los reales en el momento de ejecutar las acciones.

4.1.3 Paradigma híbrido

En el comportamiento de cualquier ser vivo, una misma acción puede tener diversos niveles en los que se originó. El movimiento de un brazo puede tener una base de razonamiento en el caso de que el cerebro ordene el movimiento para coger un objeto, o pude ser un movimiento puramente instintivo, como sería el caso de la proximidad de una llama, en el que el cerebro no ha realizado un procesamiento. De la misma manera un robot móvil debe tener ambos tipos de orientaciones. Esto hace que el paradigma híbrido, surgido en 1990 [Arkin, 1990], incorpore una capa deliberativa sobre la capa reactiva (figura 21). Con esta orientación se tiene la velocidad de actuación y cumplimiento de las restricciones temporales de respuesta del nivel reactivo, y la potencia de navegación que tiene el nivel reactivo. La línea de separación entre los niveles, depende de varios factores, aunque tiene especial importancia el uso que se hace de la información de entrada al sistema (generalmente procedente de los sensores o de otros niveles o módulos del mismo nivel, y más o menos elaborada) y la información que se requiere proporcionar a los actuadores o a otros niveles.

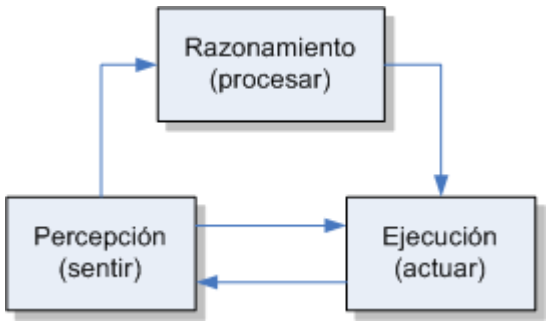

Figura 21. Paradigma híbrido.

En el planteamiento híbrido el robot mantiene la capa reactiva, ya que esta le proporciona una supervivencia básica en el entorno en que esté navegando. Sobre esa capa reactiva, se incluye una capa deliberativa que permite realizar misiones complejas. Esta capa deliberativa obtiene información del bloque de percepción de la capa reactiva, y puede interaccionar en las acciones del robot por medio de la conexión con el bloque de ejecución de la capa reactiva.

4.2 Características

Diversos autores han estado buscando las características comunes de las arquitecturas, ya sea para hacer un análisis y posterior evaluación de las mismas o para hacer una taxonomía más o menos compleja. Intentar destacar o aislar puntos comunes por los que clasificar o evaluar arquitecturas es casi imposible. Cada una de ellas tiene objetivos, componentes y funcionalidades distintas que hacen muy complicado encontrar puntos comunes. A continuación se muestran algunos de los conjuntos de características que algunos autores han realizado.

A partir de [Brooks, 1991a] y especialmente [Brooks, 1991b], Rodney Brooks, destaca algunas características específicas para los niveles de comportamientos en los que se basa su arquitectura. Entre ellas se pueden destacar.

- Generalidad (Generality) y versatilidad (Versatility) Capacidad para adaptarse a diversas situaciones.
- Racionalidad (Rationality). Actuación racional del robot.
- Capacidad para adquirir nuevos conocimientos (Ability to add new knowledge). Se refiere a la capacidad para recoger información del entorno y procesarla de manera que sea útil para la navegación del robot.
- Capacidad para aprender (Ability to learn). No se debe confundir con la característica anterior, en este caso se refiere a la capacidad de generación de nuevos comportamientos.
- Capacidad de descomposición en tareas (Taskability). Si un robot tiene la facultad de organizar una tarea en tareas menores.
- Escalabilidad (Scalability) Posibilidad de ampliación de la potencia del sistema.
- Reactividad (Reactivity) Capacidad de reacción, en el tiempo adecuado, ante los cambios del entorno.
- Eficiencia (Efficiency) Capacidad de lograr realizar las tareas optimizando unos parámetros en función del logro de unos requisitos.

En [Arkin, 1998] los criterios que se emplean para evaluar y comparar las arquitecturas son los siguientes:

- Modularidad (modularity): que trata acerca de la independencia entre los distintos componentes. Se incluyen características intrínsecas como la posibilidad de reutilización de los módulos y la escalabilidad del sistema.
- Adaptabilidad (adaptability): se entiende la capacidad de adecuar la arquitectura a las aplicaciones para las que ha sido diseñada.
- Portabilidad (portability): facilidad de hacer funcionar la arquitectura en sistemas distintos.
- Robustez (robustness): capacidad de la arquitectura de resistir los errores que puedan aparecer.

Una aproximación interesante a las características que debe tener una arquitectura para controlar la navegación de un robot móvil se encuentra en [Alami et al., 1998]. Las características que una arquitectura debe tener son:

- Capacidad de programación (programmability). Es importante que un sistema autónomo pueda desarrollar el mayor número de funciones en el mayor número de entornos posibles. La diversidad de entornos implica que la abstracción que se deba aplicar a la programación del robot se traslade a los niveles más altos de la arquitectura.
- Autonomía y adaptación (autonomy and adaptability). Cuanta más capacidad de adaptación se tenga, de más autonomía dispondrá el sistema. Es deseable que un robot tenga capacidad de adaptación ante entornos diferentes.
- Reactividad (reactivity). La capacidad de reaccionar en los límites temporales correspondientes debe ser compatible con la realización de los objetivos que el sistema tenga asignados. Lo anterior implica que el robot no debe dedicar todo el tiempo a las misiones programadas olvidando la reacción ante situaciones que pongan en peligro su propia seguridad.
- Comportamientos coherentes (consistent behavior). La arquitectura debe dar soporte a que el robot realice las tareas orientadas a los objetivos asignados y adecuadamente en el entorno en que las desarrolla.
- Robustez (robustness). La arquitectura del robot debe poder soportar la redundancia en el procesamiento de los datos. Generalmente la robustez implica en la descentralización del control, y por tanto la distribución de tareas y de datos.
- Extensibilidad (extensibility). La arquitectura debe soportar la escalabilidad del sistema, de manera que la incorporación de tareas no debe implicar cambios en el sistema. Esto lleva directamente a la capacidad de aprendizaje del sistema.

Es complicado destacar o agrupar características comunes que sirvan para evaluar una arquitectura. De todas maneras sí que se pueden obtener algunas características que dependan de la información que el sistema procesa.

A la hora de revisar distintos modelos de arquitecturas, se puede recurrir a diversas clasificaciones. En lo que respecta a las arquitecturas reactivas y las arquitecturas deliberativas, se ha optado por clasificarlas en función de la estructura de los bloques.

En el siguiente apartado se han seleccionado una serie de arquitecturas de entre la gran cantidad de arquitecturas existentes. Las arquitecturas revisadas se pueden observar en la tabla correspondiente en el anexo de arquitecturas revisadas

De entre las arquitecturas reactivas, se pueden distinguir dos filosofías muy concretas, la primera de ellas es la reactiva basada en reacciones, donde no existen comportamientos propiamente dichos, y los valores que se le pasa a los actuadores están determinados directamente por las entradas de los sensores. Aunque puede parecer extremo hablar de los vehículos de Braitenberg como arquitectura, éstos son un buen ejemplo de conexiones directas, con un procesamiento mínimo o nulo de los datos de los sensores. En otro nivel de las arquitecturas reactivas se encuentran las arquitecturas basadas en los comportamientos básicos. En este nivel está la arquitectura de subsunción de Brooks. Cabe destacar que ambos modelos están inspirados en los comportamientos de los seres vivos básicos.

En lo que a arquitecturas deliberativas se refiere, para clasificarlas se han seguido dos categorías básicas. La primera de ellas (secuencial) sigue el paradigma secuencial clásico de los sistemas deliberativos (sentir-procesar-actuar) sin intervención de niveles, un buen ejemplo es la arquitectura JPL. El segundo modelo es aquel en el que se añaden niveles (que pueden entrar incluso de lleno en una capa reactiva) deliberativos. Un buen ejemplo de este tipo es la arquitectura NASREM. Cabe destacar que ambas arquitecturas están muy relacionadas con la exploración espacial.

Organizar las arquitecturas híbridas, es quizás una labor más compleja que los dos modelos anteriores. El hecho de que haya dos niveles diferentes hace que la clasificación se pueda basar en más criterios que los dos paradigmas anteriores. En [Murphy, 2000] se hace una interesante clasificación conceptual de las arquitecturas híbridas de robots móviles, en función de diversos factores, entre los cuales caben destacar:

- ¿Cómo se distinguen las capas reactiva y deliberativa?
- ¿Cómo se organizan las responsabilidades de la capa deliberativa?
- ¿Cómo emergen los comportamientos?

Esta “clasificación” es interesante puesto que consiste más en una visión dinámica de tareas de alto nivel que en una estructuración estática de la arquitectura. A partir de esta clasificación se organizan las arquitecturas deliberativas en arquitecturas híbridas organizativas, basadas en jerarquías de estados y en orientadas a modelos. Esta clasificación se seguirá a la hora de exponer los ejemplos de arquitecturas híbridas.

4.3 Arquitecturas reactivas

4.3.1 Fundamentos de la reacción: Braintemberg

A continuación se verán las características principales de diversas arquitecturas, principalmente híbridas, desarrolladas por diversos grupos y autores.

Un ejemplo y tiene una aplicación directa para la navegación en los vehículos de Braitenberg donde las conexiones entre sensores y actuadores es directa o basada en combinaciones sencillas [Braitenberg, 1984]. En la figura 22, se puede observar un ejemplo de las conexiones directas entre sensores y actuadores.

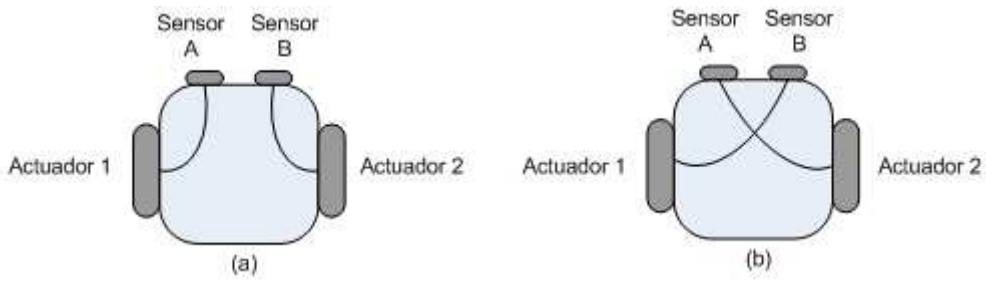

Figura 22. Vehículos de Braitemberg como aproximación a las arquitecturas reactivas.

Los vehículos de Braitenberg representan de una manera extremadamente básica los principios de navegación reactiva. Desde el punto de vista de una arquitectura, sin constituir en sí mismos una arquitectura, suponen una cota inferior, en lo que se refiere a información y estructura de las comunicaciones. A medida que se incorpora procesamiento entre los sensores y los actuadores, se va incrementando la capacidad de realización de misiones cada vez más complejas. Aunque las conexiones básicas, entre sensores y actuadores, hacen que los vehículos sean deterministas, es prácticamente imposible predecir el comportamiento del vehículo. Sin embargo a partir de la variación de las conexiones, se puede ajustar los comportamientos que el vehículo realizará.

4.3.2 Reacción basada en comportamientos: Subsumption

Aunque de difícil traducción, subsumir o categorizar, es incluir algo como un componente en una síntesis o clasificación más abarcadora, también se puede considerar como que algo como parte de un conjunto más amplio, o como caso particular sometido a un principio o norma general. Es una arquitectura no simbólica y no verbal que se usa tanto como alternativa para las arquitecturas tradicionales de Inteligencia Artificial (figura 23.a) o como módulos básicos para ampliar otras arquitecturas más complejas. La arquitectura se describe inicialmente en [Brooks, 1986]. Surge como alternativa a la aproximación secuencial de la inteligencia artificial clásica y la convierte en una aproximación de proceso paralelo (figura 23.b)

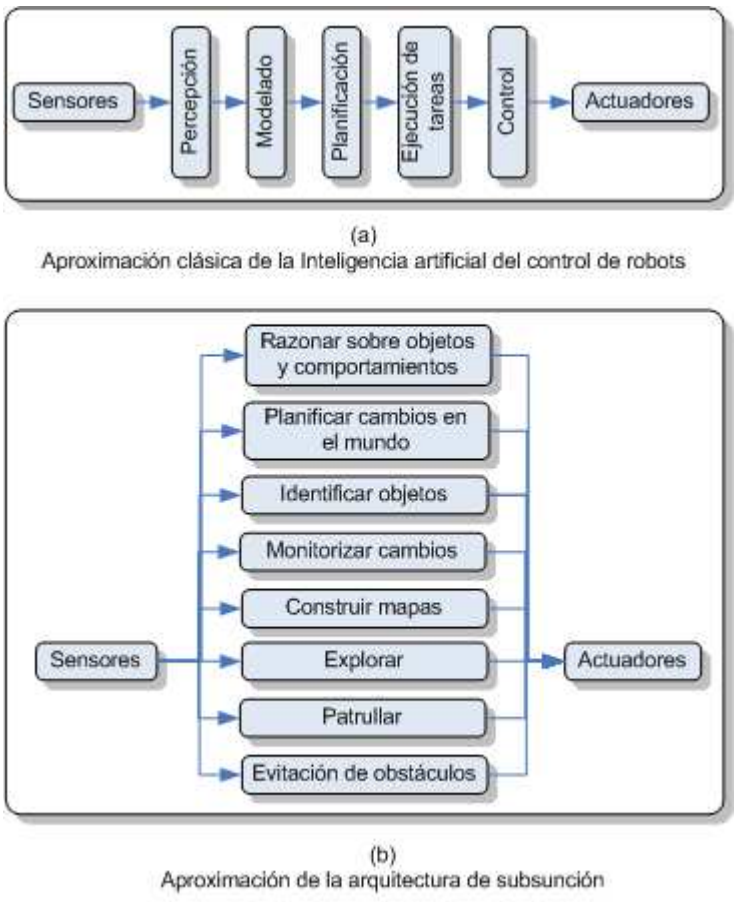

Figura 23. Visión clásica de la Inteligencia Artificial (a) y visión de la Subsunción (b).

En la pila de la arquitectura de Brooks, se mantiene una jerarquía implícita en la que los niveles más bajos se dedican a las reacciones (habitualmente se habla de comportamientos básicos) más básicas, aquellas que proporcionan supervivencia al sistema. En el caso de control de robots se suele poner como ejemplo el comportamiento la evitación de obstáculos (considerando obstáculo cualquier aspecto que ponga en peligro al robot). Si está necesidad de supervivencia básica se cumple, se pasa al siguiente nivel, en el que se tienen comportamientos, no tan prioritarios como la supervivencia, pero sí necesarios para lograr una conducta inteligente. El efecto global de los comportamientos es una respuesta inteligente del sistema ante el entorno.

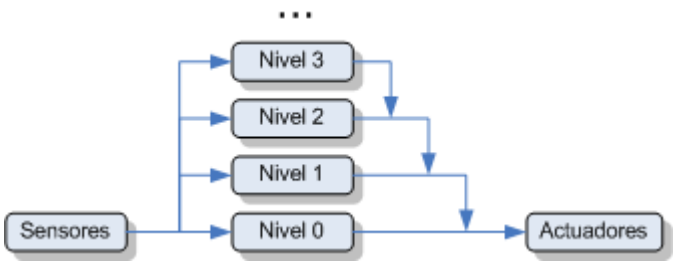

Figura 24. Fundamento de la arquitectura de Subsunción.

Evidentemente en la arquitectura de subsunción las entradas de información de cada capa son los sensores. Cada capa deberá procesar dicha información (más o menos organizada) y suministrar resultados que tendrán un efecto en los actuadores.

La implementación de cada capa se puede realizar de diversas maneras; por medio de máquinas de estado finito, tal como propone Brooks; por medio de redes neuronales, tal como se propone en [Rosemblatt and Payton, 1989].

4.4 Arquitecturas deliberativas

Evidentemente una arquitectura puramente deliberativa carece de bastante sentido en un robot autónomo, sin embargo. Para encontrar una aproximación deliberativa se debe buscar en los vehículos donde cada paso debe medirse con extremo cuidado, generalmente vehículos de exploración espacial.

4.4.1 Modelo secuencial: JPL

En [Gat, 1990] se expone una arquitectura a la que se le referencia por JPL (Jet Propulsion Laboratory) Architecture. Esta arquitectura se propuso con el fin de proveer una autonomía parcial a un vehículo de exploración planetaria. La arquitectura se componen de cuatro módulos principales (figura 25): el módulo de percepción, el planificador de caminos, el Planificador y Monitor de Ejecución y el sistema Ejecutor.

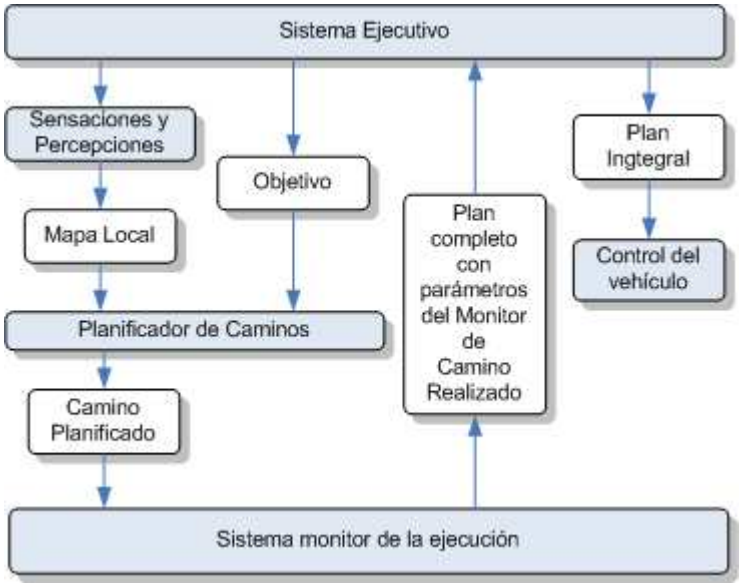

Figura 25. Arquitectura JPL para el control de la navegación robots.

La principal tarea del módulo de percepción es crear un mapa local del entorno, usando los datos de los sensores y datos globales enviados por el módulo orbitador, lo que implica que se necesita el apoyo de otro vehículo que le proporciona una información, posiblemente a dicha se le podría considerar un mapa global del entorno. El mapa local creado es empleado por el planificador de caminos y por el Monitor del Camino Realizado para planificar un camino libre de colisiones (de aproximadamente de 10 metros) y un plan completo, incluido los parámetros del monitor del camino realizado. Este plan de movimientos se obtiene simulando el desplazamiento del robot por el camino libre de colisiones que se haya planificado, y anticipando los posibles fallos y los datos sensoriales que se deben adquirir por parte del monitor. El plan de movimientos parametrizado es usado finalmente por el sistema ejecutivo para monitorizar la ejecución de las tareas del plan. En la práctica sólo se pueden definir "acciones reflejas" simples (por ejemplo, detener el robot y moverse atrás a una posición segura).

Esta arquitectura fundamentalmente aplica un paradigma secuencial SMPA (Sense-Model-Plan-Act) ya que ejecuta maniobras predefinidas cuando se detectan situaciones peligrosas. Esta arquitectura representa una extensión de un nivel mínimo de integración de un componente reactivo en una arquitectura deliberativa.

4.4.2 Modelo paralelo: NASREM

NASREM son las siglas de NASA Standard REference Model (modelo estándar de referencia de la NASA). Como su nombre indica, es un modelo teórico, pero no una implementación. El modelo surge como una arquitectura de un sistema RCS (Real-Time Control System) [Albus, 1991] y se desarrolla como referencia estándar de la NASA para robots en [Albus, 1993]. Para ser implementado como sistema real, los elementos inteligentes de la arquitectura deben integrarse en módulos de computación. La interconexión de los módulos, se lleva a cabo a partir de redes y jerarquías de interconexión. Como todo sistema inteligente, un sistema de procesamiento de los datos proporcionados por los sensores mantiene un modelo interno del mundo real que rodea al robot. Los comportamientos se generan por medio de acciones de control sobre los actuadores para lograr una serie de objetivos que se deben encontrar en el contexto del modelo de mundo que se haya percibido. El modelo consistente en seis elementos básicos:

- Actuadores.
- Sensores.
- Procesamiento sensorial.
- Modelo del mundo.
- Generador de comportamientos.
- Estimador de Valores (Value Judgment).

Estos elementos se integran en un modelo jerárquico que se puede ver en la figura 26.

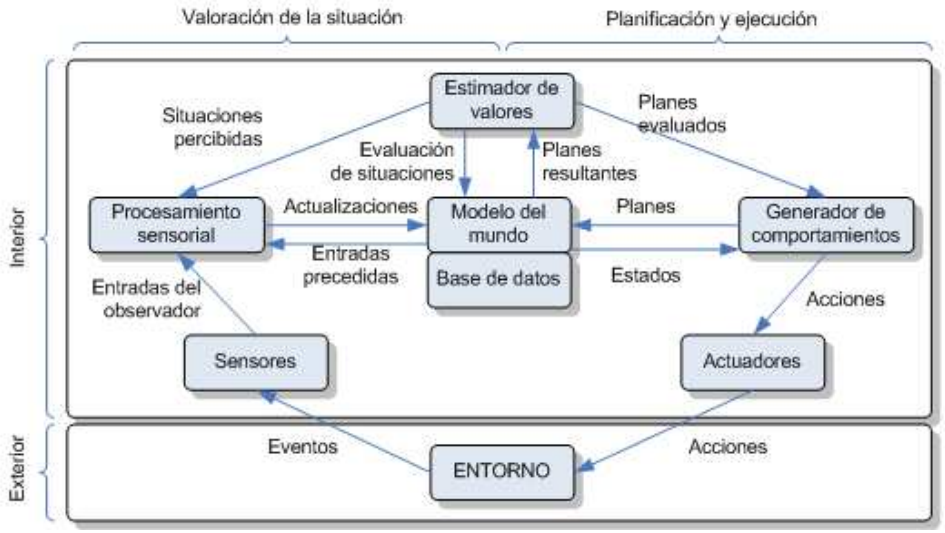

Figura 26. Arquitectura NASREM de navegación de robots.

Los actuadores son las salidas del sistema inteligente. Un sistema puede tener desde muy pocos actuadores hasta cientos de ellos, y todos deben ser coordinados para que puedan desarrollar las tareas correspondientes. Los sensores son la entrada al sistema inteligente, y pueden ser empleados para monitorizar tanto el entorno como para conocer el estado interno del robot. El sistema de procesamiento sensorial es el que proporciona la posibilidad de tener percepción al robot. Este sistema compara lo que se está observando con lo que se espera a partir del modelo interno del mundo. A partir de una gran cantidad de sensores y de mediciones a lo largo del tiempo, la información que proporcionan puede ser fusionada para generar un modelo del mundo que sea útil aunque carezca de la exactitud que el sistema puede proporcionar.

El modelo NASREM tiene seis capas y está organizado en tres columnas: procesamiento sensorial, modelado del mundo y descomposición de tareas (figura 27).

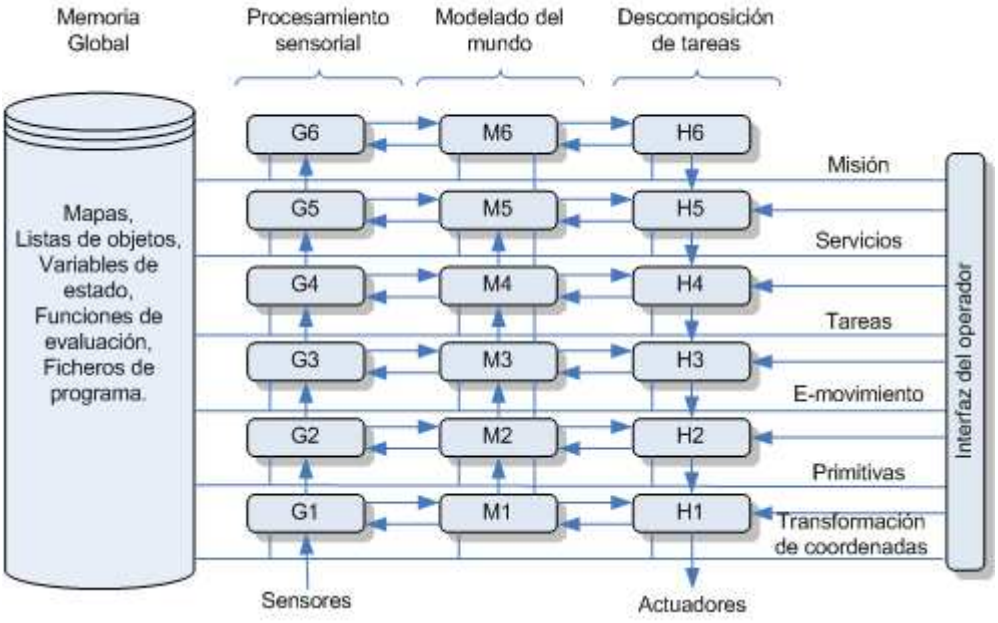

Figura 27. Componentes de control de la arquitectura NASREM.

La resolución temporal del modelo está en relación a la capa de la que se trate, de forma que, por ejemplo, en la capa 1 (servo) el intervalo de reloj está del orden de milisegundos, mientras que en la capa superior (misión), el intervalo de reloj es del orden de cuatro segundos.

Se ha considerado el modelo NASREM como modelo deliberativo, por el ámbito de conocimiento con el que trabaja, especialmente en las capas superiores. Es posible que el primer nivel (el más cercano al hardware) se pueda considerar perfectamente un nivel reactivo, pero el hecho de que en todos los niveles exista una representación del mundo, permite catalogar el modelo como un modelo reactivo.

La evolución de la arquitectura puede verse en [Albus and Barbera, 2005], donde se puede observar la interesante forma de abordar jerárquicamente la generación de los mapas de navegación en función de las necesidades.

4.5 Arquitecturas híbridas organizativas

Consisten en las arquitecturas que realizan una división de responsabilidades, de la misma forma que se hace con los negocios. Entre estas se encuentran las arquitecturas AuRA [Arkin, 1990] o la SFX [Murphy, 2000]. Las arquitecturas gerenciales distinguen claramente entre capa reactiva y deliberativa, siendo la capa deliberativa aquella que tiene un conocimiento global del mundo, mientras que la capa reactiva es la que contiene los comportamientos básicos con algo de memoria o percepción y un estado externo. En la capa deliberativa se organizan las responsabilidades por medio de una jerarquía de responsabilidades. Abstrayendo el concepto de arquitectura gerencial al campo de la información, las arquitecturas gerenciales tienen su ámbito en la información espacial.

4.5.1 AuRA

Aura es el acrónimo de Autonomous Robot Architecture (Arquitectura para robot móvil), introducida por Ronald Arkin [Arkin, 1989], [Arkin, 1990]. Es una arquitectura específica para robots móviles basada en la teoría de esquemas y con una fuerte inspiración en los sistemas biológicos y enfoques psicológicos y neurofisiológicos de comportamientos. La arquitectura distingue dos capas o zonas muy concretas de actuación: la capa deliberativa y la capa reactiva, los componentes de ambas capas pueden observarse en la figura 28.

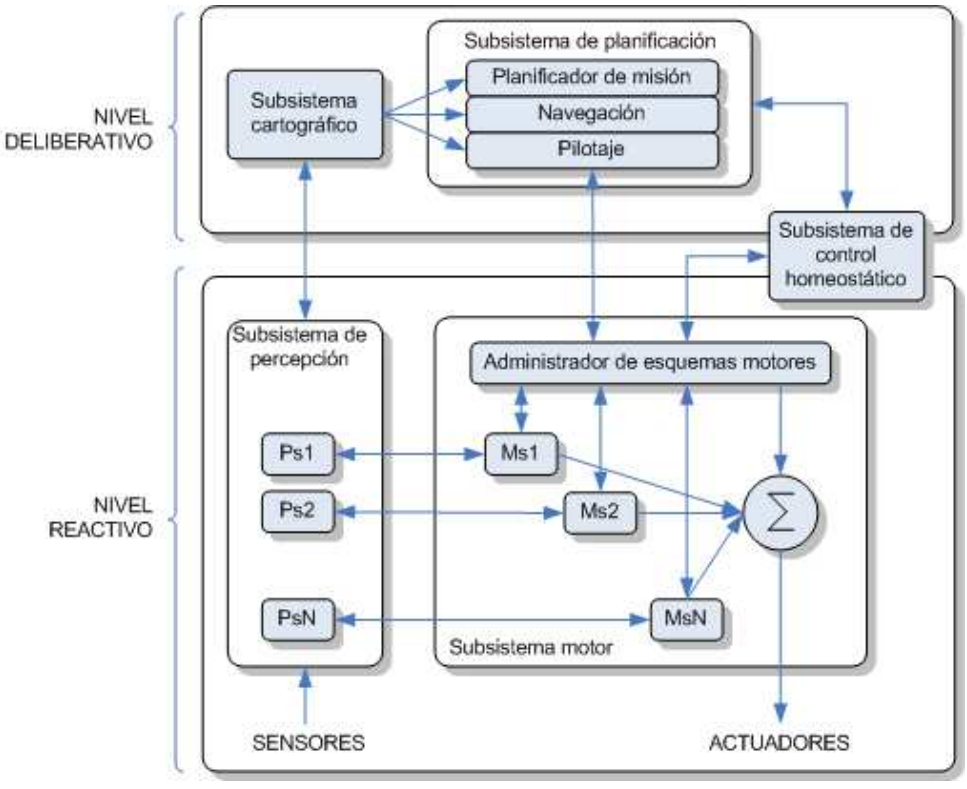

Figura 28. Arquitectura de control AuRA para la navegación de robots.

El control en el nivel reactivo se realiza a partir de comportamientos primitivos (esquemas), la composición de los comportamientos simples dan lugar a comportamientos más complejos. La complejidad de los comportamientos dependerá de la complejidad de la combinación de esquemas que se realice. Los subsistemas que componen este nivel son:

- Subsistema de percepción: es el responsable de adquirir, por medio de los esquemas de percepción (Ps) la información proporcionada directamente por los sensores, filtrarla y enviarla tanto al subsistema cartográfico (del nivel deliberativo) como al subsistema motor (nivel reactivo). Este subsistema tendrá una librería con todos los esquemas de percepción disponibles.
- Subsistema motor: es el encargado de gestionar los esquemas motores (Ms), para ello deberá contener una librería con todos los esquemas motores disponibles. El administrador de esquemas motores accede tanto a los esquemas de percepción como al nivel deliberativo para obtener los esquemas de los comportamientos seleccionados. Un mismo comportamiento puede estar compuesto de varios esquemas motores que serán coordinados por máquinas de estados finitos. Dentro del subsistema motor se calculará la contribución de cada esquema motor al movimiento total, realizando la composición entre los mismos.

Entre los niveles reactivo y deliberativo (aunque más cercano al reactivo), se encuentra un componente de vital importancia, ya que el subsistema homeostático:

- Subsistema homeostático. Este subsistema tiene una labor muy importante, ya que involucra las tareas de supervivencia del robot en circunstancias peligrosas. El control homeostático es el responsable de mantener un estado de equilibrio estable entre los diferentes elementos de la arquitectura. Este control debe permitir al robot alterar dinámicamente su comportamiento en los niveles inferiores. El robot debe ser capaz de alterar su propio comportamiento tanto en función de su entorno (responsabilidad del subsistema de percepción) como en función de su estado interno, de ahí el nombre de homeostático. Para que el sistema homeostático pueda realizar correctamente su labor son necesarios dos esquemas:
    - Esquemas transmisores que se asocian a los sensores internos del robot, y transmiten al subsistema motor sus valores.
    - Esquemas receptores, que reciben la información de los esquemas transmisores y la proporcionan a los esquemas motores a los que estén asociados.

El nivel deliberativo lo componen el subsistema cartográfico y el subsistema de planificación.

- Subsistema cartográfico: gestiona el razonamiento espacial, se compone de una memoria a largo plazo, con las condiciones e información fija de la navegación (mapas globales) y de una memoria a corto plazo donde se tiene un modelo del entorno más cercano y variable, formado a partir de la información de los sensores.
- Subsistema de planificación: este subsistema es el responsable de la navegación deliberativa. Se compone de tres módulos.
    - Módulo de misiones. Responsable de la planificación de alto nivel. Determina todos los datos necesarios para que se de el comportamiento global esperado.
    - Módulo de navegación. Es el módulo que debe generar la trayectoria global que guiará al robot en la navegación deliberativa.

- Módulo de pilotaje. A partir de la trayectoria definida en el módulo de navegación, selecciona los comportamientos que debe realizar y genera la serie de esquemas motores correspondientes. Una vez seleccionados los comportamientos necesarios, se pasan parametrizados al subsistema motor. Los esquemas elegidos llevan asociados unos esquemas de percepción concretos que proporcionan la información necesaria para que el esquema motor pueda funcionar correctamente. El entorno puede hacer variar los esquemas de percepción necesarios para un esquema motor

El hecho de que un esquema motor pueda variar, de forma dinámica, los esquemas de percepción necesarios hace que el hecho de elegir un esquema adecuado pueda variar el efecto en la navegación. Para poder optimizar este aspecto, se deben almacenar unos hechos que permitan tener un criterio sobre la elección de uno u otro esquema. Esta base de hechos se compone a partir de información del entorno, condiciones internas del robot y requerimientos concretos de la misión. La elección de los esquemas se realiza por medio de unas reglas heurísticas sobre la anterior base de hechos. Los esquemas son un buen ejemplo de consumidores y generadores de información. En la figura 29, se observa cómo están asociados los esquemas.

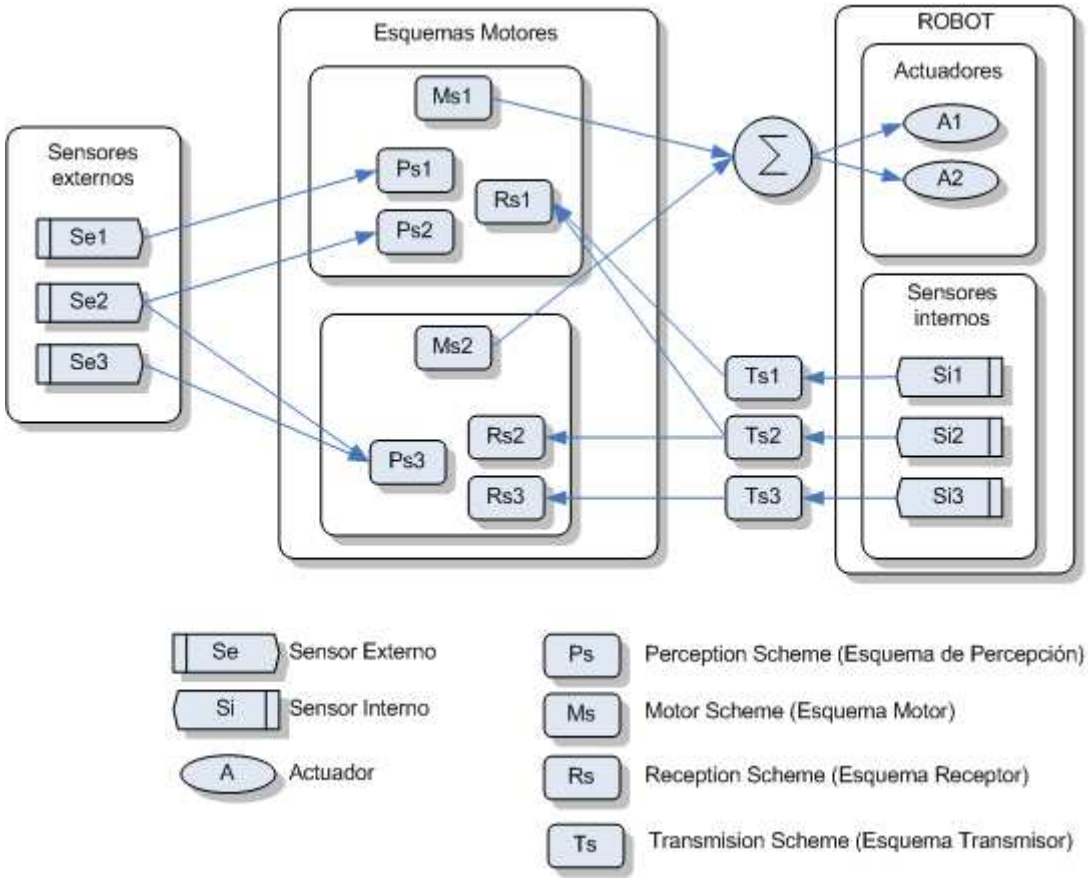

Figura 29. Relación funcional de los elementos de la arquitectura AuRA.

AuRA tiene un diseño muy modular con módulos muy bien diferenciados, lo que permite que unos módulos puedan ser reemplazados por otros sin problemas, lo que permite una flexibilidad muy grande a la hora de poder ser implementada.

4.5.2 SFX

Desarrollada por Robin Murphy [Murphy, 2000], la arquitectura SFX (Sensor Fusion Effects) puede considerarse una extensión de la arquitectura AuRA, donde se añade una capa de robustez con la incorporación de procesos de fusión sensorial y la inclusión de la tolerancia a fallos (figura 30). La información sensorial está disponible tanto en el nivel reactivo como en el deliberativo mediante el uso de estructuras de datos tipo pizarra [Nii, 1989], comunes en numerosos sistemas de Inteligencia Artificial para formar estructuras de conocimiento global [Murphy and Arkin, 1992].

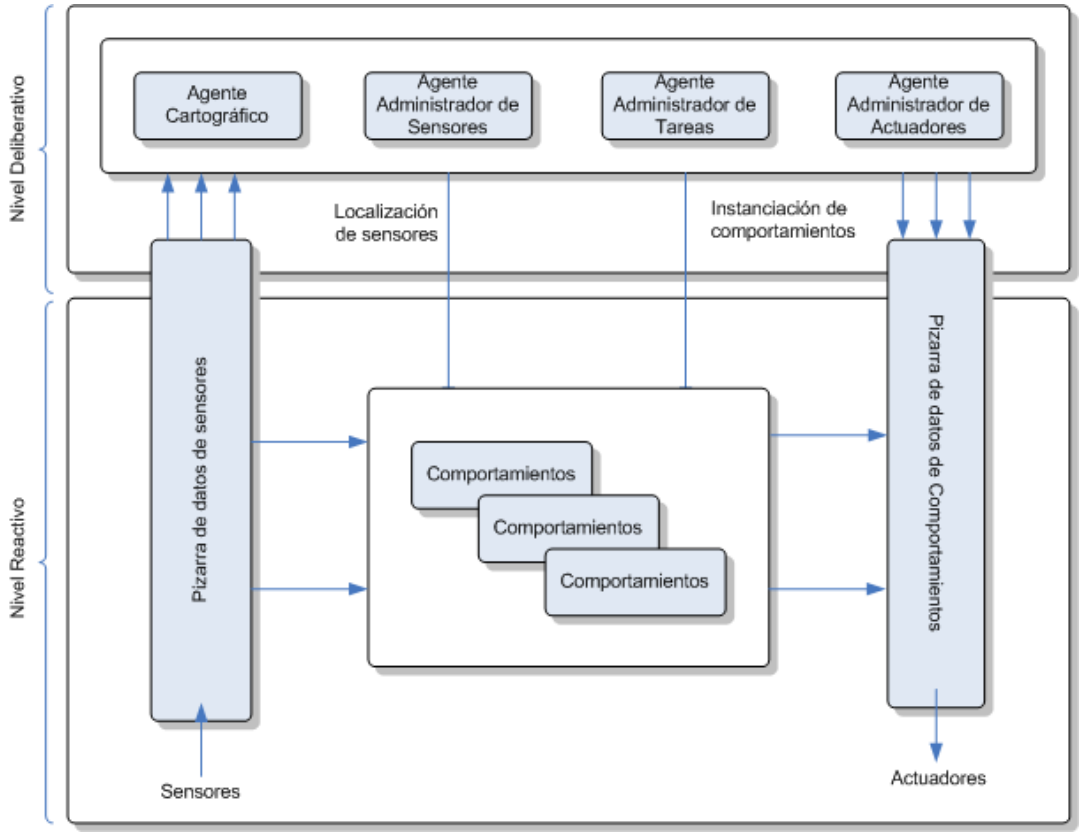

Figura 30. Arquitectura de control de robots SFX.

El nivel deliberativo está formado por componentes software independientes denominados agentes que están especializados en determinadas funciones y pueden interactuar entre ellos. El agente principal es el Planificador de misiones, interactúa con el usuario y especifica las directrices de la misión a los otros agentes del nivel deliberativo. El objetivo de los agentes es encontrar los comportamientos necesarios que aseguren la realización de la misión cumpliendo con las restricciones establecidas. Asimismo, existen agentes para determinar los sensores que utilizarán los esquemas de percepción y esquemas motores de los distintos comportamientos. Si un sensor fallara, estos agentes podrían averiguar la causa del fallo e intentar solucionarlo o podrían decidir el uso de algún sensor alternativo mediante la incorporación, en el comportamiento, del esquema de percepción correspondiente.

El nivel reactivo se divide en dos capas identificadas por los comportamientos estratégicos y por los comportamientos tácticos. A diferencia de la arquitectura AURA, donde se utilizan campos potenciales para combinar los comportamientos, SFX utiliza un método de filtrado donde algunos comportamientos fundamentales inhiben a otros comportamientos.

La idea es similar al proceso realizado en la arquitectura reactiva Subsumption de Brooks [Brooks, 1986] pero en este caso los comportamientos de la capa inferior (comportamientos tácticos) inhiben a los comportamientos de la capa superior (comportamientos estratégicos). Por ejemplo, la velocidad emergente en la arquitectura AURA es siempre el resultado de la suma ponderada de todos los comportamientos activos, mientras que en la arquitectura SFX existe un control de la velocidad máxima a través de un comportamiento táctico específico. Esto quiere decir que los comportamientos estratégicos de la capa superior proporcionan una velocidad al igual que ocurre en la arquitectura AURA y, después, el comportamiento táctico de la capa inferior filtra este valor dependiendo de si es superior al valor máximo o no. La velocidad resultante siempre será el valor menor de ambos. FSX basa esta técnica en la idea de que la mayoría de las tareas pueden expresarse mediante un solo comportamiento estratégico y uno o varios tácticos que actúen de filtros del estratégico.

4.5.3 LAAS (HILARE)

La arquitectura HILARE, desarrollada en el LAAS (Laboratoire d'analyse et d'Architecture des Systèmes, Toulouse) y especificada [Alami et al., 2000], se compone de tres niveles jerárquicos, con diferentes restricciones temporales y con manipulación de una distinta representación de datos (figura 31).

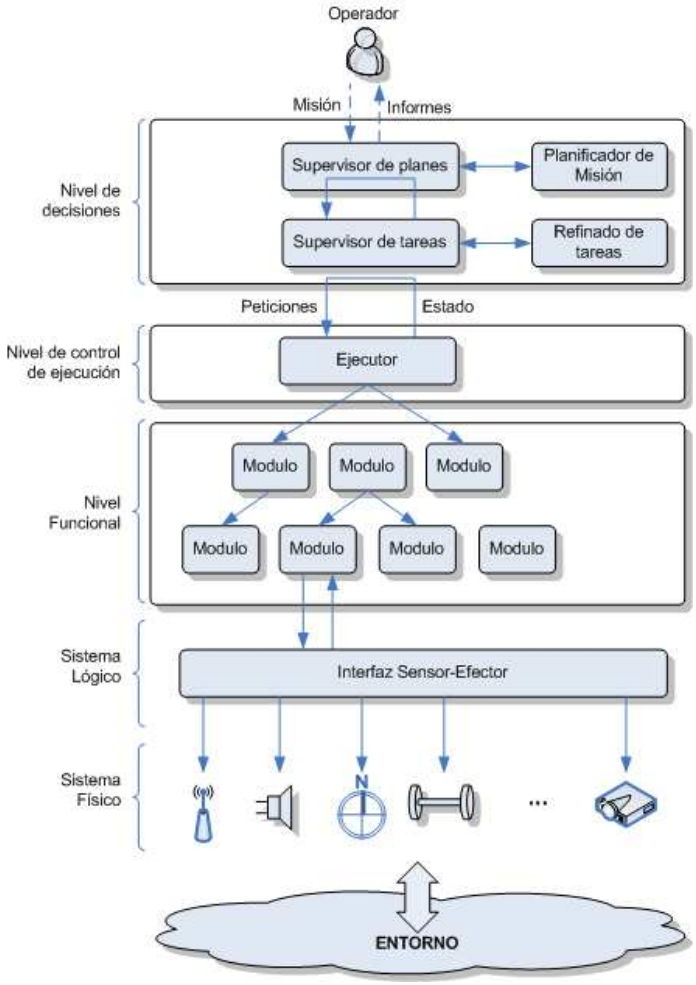

Figura 31. Arquitectura LAAS de control de la navegación de robots.

El nivel más bajo o cercano al hardware, y consiguientemente a la navegación reactiva, es el funcional; este nivel incluye las capacidades básicas del robot de acción y percepción. Estas capacidades están encapsuladas en unos módulos comunicados controlables. El siguiente nivel es el nivel de ejecución o ejecutivo. Este nivel, controla y coordina dinámicamente la ejecución de las funciones distribuidas en los módulos, de acuerdo con las tareas que se especifican en el nivel superior. El último de los niveles es nivel de decisión. En este nivel se incluyen las capacidades para producir las tareas del plan y supervisar su ejecución. Este nivel puede tener una gran cantidad de capas, de acuerdo con las capacidades que se le requieran. Realmente es un planificador y supervisor que permite integrar la reactividad y la deliberación.

4.6 Arquitecturas híbridas basadas en jerarquías de estados

Son arquitecturas en las que las actividades están organizadas estrictamente en el ámbito temporal, también llamado "estado de conocimiento". Como ejemplo de esta arquitectura está la 3T [Bonasso, 1997]. Normalmente son arquitecturas que tienen tres capas, dependientes del instante temporal en la que trabajan. Las capas son, evidentemente, la dedicada al futuro, al pasado y al presente. Estas arquitecturas distinguen la capa reactiva de la deliberativa, en función del instante temporal en la que trabajan. La capa deliberativa trabaja con el conocimiento pasado y futuro, aunque más que conocimiento futuro convendría hablar de suposiciones o predicciones. La organización de las responsabilidades de la capa deliberativa se realiza por medio del estado interno temporal, el estado futuro es responsabilidad del planificador y el estado pasado, del secuenciador, se podría asumir que la monitorización proporciona la información pasada, mientras que el planificador, genera la información futura. En este tipo de arquitecturas, la velocidad de ejecución también proporciona un modo de organizar las tareas (con lo que supone la planificación en tiempo real). Los comportamientos emergen por medio de la generación y la monitorización de secuencias de comportamientos, la unión de comportamientos llamadas "habilidades" (skills) y la subsunción (subsumption). Abstrayendo el concepto de arquitectura basada en jerarquía de estados al campo de la información, estas arquitecturas tienen su ámbito en la información temporal.

4.6.1 3T (three tiered)

La arquitectura 3T (Three Tiered) es una arquitectura híbrida. Los primeros orígenes se encuentran en [Bonasso, 1995a] y está especificada en, [Bonasso, 1995b] y [Bonasso, 1997].Se trata de una arquitectura diseñada para funcionar en diversos tipos de robots de forma independiente del hardware. Usa varios niveles de abstracción que permiten a un operador ordenar a un robot la realización de varias tareas. La arquitectura está diseñada para funcionar en entornos dinámicos. Los componentes pueden verse en la figura 32.

Originalmente, en el nivel superior se encuentra el planificador, que permite seleccionar objetivos. Estos son descompuestos por el nivel de secuencias en una serie de acciones de bajo nivel. Estas acciones vuelven a ser descompuestas en los llamados "skills" o habilidades por el nivel más bajo.

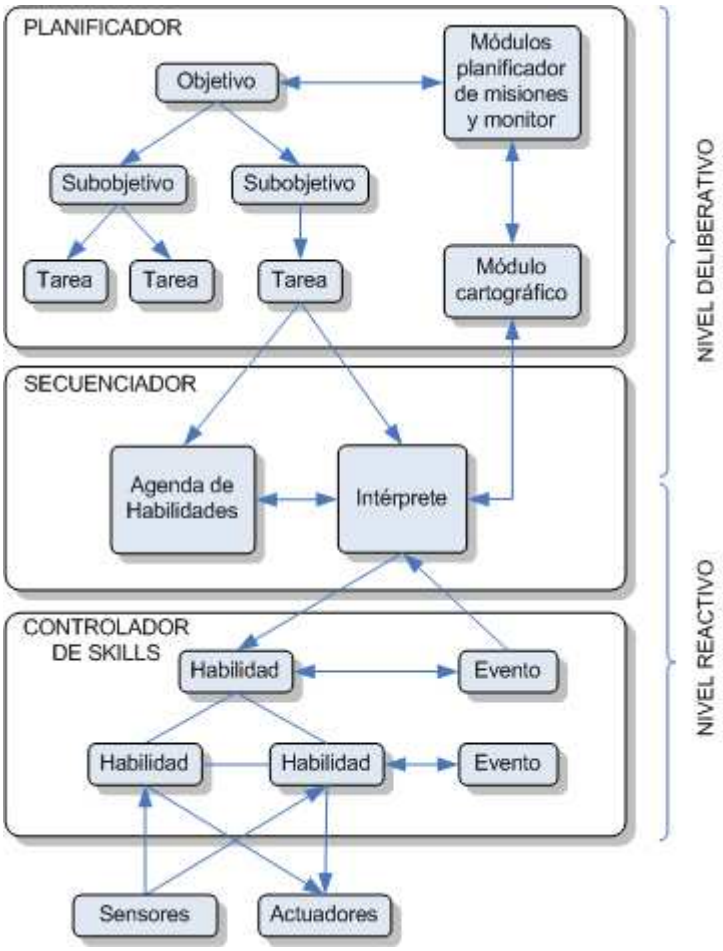

Figura 32. Arquitectura de control inteligente 3T.

Esta estructura de tres capas, se puede adaptar perfectamente a un paradigma híbrido [Bonasso, 1997], si se tiene en cuenta que la capa del planificador es una capa deliberativa, y la capa de gestión de habilidades es una capa reactiva, si estas habilidades son reactivas. Un sistema de secuenciado que se localiza entre las dos capas anteriores se encarga de activar o desactivar conjuntos de habilidades reactivas.

4.6.2 ATLANTIS

La arquitectura ATLANTIS [Gat, 1992] es muy similar a la 3T. Sus autores la definen como una arquitectura heterogénea y asíncrona basada en un modelo de acciones [Gat, 1991]. Uno de los principios básicos que trata esta arquitectura es no emplear el modelo de estados-acciones clásico de la inteligencia artificial.

El modelo de estado-acción se basa en la idea de que la evolución temporal de la configuración de un sistema (o un entorno) puede ser descrito como una secuencia de estados discretos. Un estado se transforma en otro por medio de una acción, que puede ser tanto una estructura computada o una acción física del sistema. Normalmente a la computación que se hace para que de cómo consecuencia una acción que cambie de estado se le suele llamar operación u operador. ATLANTIS está basado en un modelo de acciones continuo. Este modelo está basado en operadores cuyo tiempo de ejecución es insignificante en comparación de costosos procesos que puedan ejecutarse. Los procesos se llaman actividades y los operadores que los inician y terminan se conocen como decisiones.

Una decisión puede inicializar una actividad que, a su vez, inicie otras actividades, si se asume la ausencia de ciclos en la red que forman las actividades y las decisiones, se puede considerar al conjunto de actividades de una forma jerárquica.

Los niveles altos de actividades contienen procesos computacionales que van produciendo más actividades en niveles inferiores. La parte más inferior del modelo jerárquico la componen las actividades primitivas. Estas actividades primitivas son procesos sensorio-actuadores del más puro nivel reactivo. Debido a que no hay una correspondencia entre decisiones y cambios en el mundo, un modelo de acciones basado en actividades es difícil de analizar como un modelo de estados-acciones. La arquitectura ATLANTIS la forman tres componentes (figura 33), un controlador, un secuenciador y un componente deliberativo.

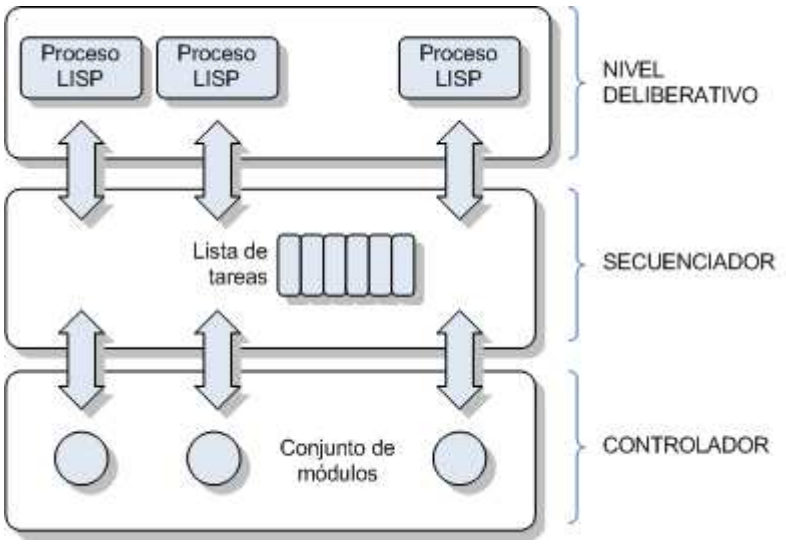

Figura 33. Arquitectura ATLANTIS de control inteligente.

El controlador es el componente responsable de gobernar las actividades primitivas, es decir las actividades reactivas, estas actividades se caracterizan por no tener que realizar ninguna acción computacional. Aunque la primera aproximación es diseñar esta capa basándose en la teoría de control clásica, en muchos casos no es posible debido a la gran dificultad de realizar un modelo matemático del entorno complejo. Se debe tener en cuenta que cuanto más complejo se considera el entorno, más se justifica un paradigma deliberativo, o un nivel computacional (más fácil de programar que un sistema de control). En los casos en los que no se puede utilizar el control matemático, la arquitectura debe proporcionar un entorno que permita ejecutar algoritmos de control. En el caso de ATLANTIS, los autores han diseñado un lenguaje de programación llamado ALFA. Los programas realizados en ALFA se componen de módulos computacionales que están conectados entre ellos por medio de canales de comunicaciones. Los módulos pueden ser insertados o eliminados sin tener que reestructurar la red de comunicaciones.

El secuenciador es el componente responsable de controlar la secuencia de actividades primitivas y las computaciones deliberativas. El secuenciar es una tarea complicada, debido a que repartir efectivamente las acciones. Es importante tener en cuenta que se debe mantener internamente una buena cantidad de información de acciones anteriormente planificadas para decidir qué acciones se deben tomar. El hecho de errar en la secuenciación, hace que se puedan trasladar dichos errores a las acciones secuenciadas.

El principio que permanece por debajo del secuenciador es el del "fallo consciente", es decir un fallo en el que el sistema, de alguna manera, puede darse cuenta del mismo. Esto hace que se puedan crear algoritmos que puedan fallar. El secuenciador inicia y termina las actividades primitivas activando o desactivando conjuntos de módulos en el controlador. El secuenciador puede enviar parámetros al controlador por medio de los canales de comunicaciones. El progreso de una actividad primitiva es monitorizado examinando los canales.

El secuenciador está diseñado basándose en el paquete RAP (Reactiva Action Package) de Firby, aunque sin llevar exactamente la misma filosofía. El sistema mantiene una cola de tareas, la cual sólo es una lista de las tareas que el sistema debe realizar. Cada tarea contiene una lista de métodos para llevar a cabo las tareas, junto con las anotaciones que describen ajo qué circunstancias cada método es aplicable. Un método es cualquiera de las acciones primitivas o una lista de sub-tareas que se instalan en la lista de tareas. Cuando una tarea falla, se ejecuta un método alternativo. El secuenciador de ATLANTIS no es exactamente el mismo que el RAP, ya que en ATLANTIS se consideran actividades, mientras que en RAP se secuencian acciones atómicas. Esto implica que es sistema debe procurar que no se inicien dos actividades que puedan interferirse, además interrumpir una actividad implica que debe dejar el sistema, pero asegurándose que su interrupción no va a bloquear a otras actividades.

El componente deliberativo se encarga de realizar las tareas computacionales "consumidoras de tiempo" como la planificación y el mantenimiento de modelos del mundo. El componente deliberativo lleva a cabo las tareas de computación bajo el control del secuenciador, por lo que todos los procesos deliberativos son iniciados, o terminados, por el secuenciador. No hay limitaciones en cuanto la estructura de computación, excepto que puedan ser iniciadas o finalizadas por el secuenciador. Habitualmente consiste en un conjunto de programas en LISP que implementan los algoritmos tradicionales de inteligencia artificial.

Tanto el componente secuenciador como el componente controlador tienen la misión de proveer un interfaz que conecte los sensores y actuadores físicos con un sistema que ejecute algoritmos de inteligencia artificial clásico. La arquitectura ATLANTIS proporciona una conexión entre un modelo tradicional de inteligencia artificial basado en estados y acciones con un modelo basado en actividades continuo.

4.7 Arquitecturas híbridas orientadas a modelos

Se entienden como arquitecturas orientadas a modelos aquellas en las que los modelos sirven como sensores virtuales. Entre las arquitecturas que se clasifican de este tipo, están: Saphira [Konolige, 1998] y TCA [Simmons, 1994]. La capa deliberativa contiene todo lo relativo a comportamientos que conlleven lograr un objetivo, la capa reactiva contiene los comportamientos como pequeñas unidades de control, estas unidades operan en el presente, pero pueden usar el conocimiento global en forma de sensores virtuales.

La organización del sistema se hace por medio de componentes reactivos y deliberativos, un modelo del mundo y del estado del robot, esto supone una vuelta al paradigma jerárquico con un modelo del mundo y con sensores virtuales. Los comportamientos emergen por medio de la secuencia de comportamientos, por combinación en lógica borrosa de comportamientos o por voto. Abstrayendo al concepto de información, las arquitecturas orientadas a modelos tienen su ámbito en la información tanto temporal como espacial.

4.7.1 Saphira

La arquitectura Saphira [Konolige, 1998] es un sistema integrado de sensorización y control para las aplicaciones robóticas. Como componente central se encuentra el LPS (espacio de percepción local), una representación geométrica del entorno existente alrededor del robot. Debido a que diferentes tareas necesitan distintas representaciones, el LPS está diseñado para acomodarse a varios niveles de interpretación de la información sensorial.

El LPS proporciona al robot conocimiento sobre su entorno y es fundamental para las tareas de fusión de información, planificación del movimiento y la integración de la información del mapa. La arquitectura de control y percepción hace constante referencia al LPS, el cual proporciona a la arquitectura coherencia en la representación.

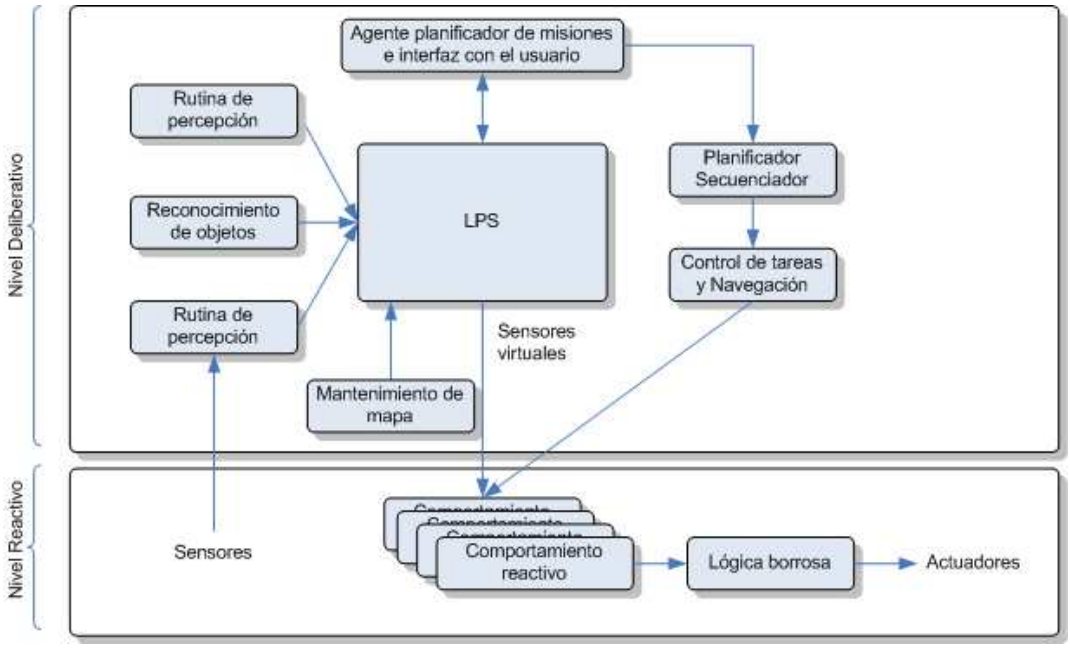

Figura 34. Componentes de la arquitectura de control Saphira.

La organización, como se puede observar en la figura 34, es parcialmente vertical y parcialmente horizontal. La organización vertical se puede observar tanto en la percepción (lado izquierdo), como en la acción (lado derecho). Varias rutinas de percepción se encargan de añadir información sensorial al LPS y de procesarla para producir información que puede ser utilizada para el reconocimiento de los objetos y la navegación. Los comportamientos más complejos que realizan acciones dirigidas por objetivos son utilizados para guiar el comportamiento reactivo. En el nivel de tareas, los comportamientos son secuenciados y su progreso es monitorizado a través de eventos en el LPS. La organización horizontal aparece porque los comportamientos pueden elegir la información apropiada del LPS. Los comportamientos críticos en el tiempo, como evitar obstáculos, confían más en un procesamiento simple de los sensores al estar disponible rápidamente.

En el nivel de control, la arquitectura Saphira se basa en el comportamiento: el problema de control se descompone en pequeñas unidades de control llamadas comportamientos básicos (como evitar obstáculos o seguir un pasillo).

Una de las características distintivas de Saphira es que los comportamientos están escritos y combinados utilizando técnicas basadas en la lógica borrosa. Los diferentes componentes de la arquitectura son independientes y no tienen por qué ejecutarse en el mismo nodo correspondiente al robot que están controlando.

4.7.2 TCA

La arquitectura TCA (Task Control Architecture) se describe en [Simmons, 1994], se basa en el concepto de control estructurado. Esta aproximación parte de componentes deliberativos básicos a los que se les enlazan comportamientos reactivos. El resultado es un entorno que combina comportamientos deliberativos y reactivos. El término de control de tareas (Task Control) de la arquitectura, se refiere al problema de coordinar componentes de percepción, planificación y ejecución en un robot para lograr una serie de objetivos. Las tareas de control incluyen los problemas de determinar qué objetivos deben atenderse, construir planes (con cualquier nivel de detalle), monitorizar el progreso y realizar las transacciones correspondientes.

Realmente TCA no proporciona unos comportamientos concretos para unas tareas concretas, sino que proporciona un sistema operativo para controlar un robot, así como los mecanismos para las comunicaciones distribuidas, descomposición de tareas o gestión de los recursos del robot. Un robot construido usando TCA, consiste en diversos módulos de tareas específicas que se comunican por medio del envío de mensajes por medio de un denominado Módulo Central de Control. Los módulos que proporciona la arquitectura son los que se conocen como construcciones de control para realizar comportamientos tanto deliberativos como reactivos. Las construcciones de control están diseñadas para integrarse sin problemas, e incluyen soporte para:

- Comunicaciones distribuidas entre procesos.
- Descomposición de tareas y restricciones temporales entre sub-tareas.
- Gestión y uso de recursos.
- Monitorización de la ejecución.
- Manejo de excepciones.

En el nivel básico, TCA soporta comunicaciones y procesamiento distribuido. El sistema de un robot que emplea la arquitectura TCA consiste en una serie de módulos específicos del robot (programados en C o LISP) y el módulo central de control que es común a todos los sistemas que usan TCA. Un ejemplo de cómo se comunican los componentes se tiene en la figura 35, donde se muestran los módulos del sistema de desplazamiento del robot Amber.

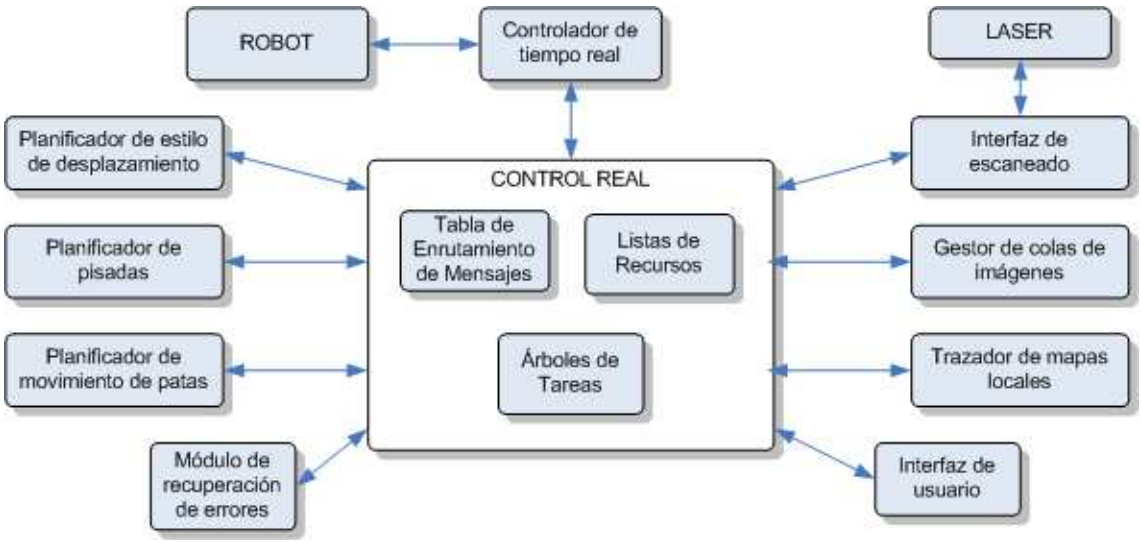

Figura 35. Componentes de la arquitectura TCA de control inteligente.

Los mensajes no son interpretados por el módulo central. Los tipos de mensajes que proporciona TCA son los siguientes. Cada uno de los tipos puede tener algunas diferencias semánticas con el resto.

- Mensajes de información: son mensajes pasados en una sola dirección, usados para pasar información de forma asíncrona entre módulos.
- Mensajes de preguntas: son mensajes bi-direccionales en los que se envía una réplica al módulo que lo haya enviados. Estos mensajes son bloqueantes hasta que el receptor reciba la información.
- Mensajes objetivos. Estos mensajes se emplean para descomponer una tarea en sub-tareas.
- Mensajes de comandos. Son mensajes similares a los mensaje objetivo, pero que representan acciones ejecutables por el robot.
- Mensajes de monitor. Son mensajes usados para monitorizar la acción de la ejecución.
- Mensajes de excepción: son mensajes que se usan para atender a situaciones excepcionales.

En el corazón de la arquitectura TCA se encuentra una representación jerárquica de tareas y sub-tareas. Realmente el Módulo Central de Control no es más que un gestor de mensajes entre componentes, y son los componentes con las prioridades entre los mensajes los que confieren la estructura de arquitectura. Esta estructura jerárquica es dinámica, lo que le confiere una cierta potencialidad.

4.8 Arquitecturas híbridas organizadas en niveles

A continuación, y debido a su interés se exponen otros modelos de arquitecturas híbridas para el control de robots móviles. Estas arquitecturas no son fáciles de clasificar en alguno de los grupos anteriores, pero tienen en común que se organizan en diferentes niveles bien diferenciados que no dependen de características temporales, como la organización empleada.

4.8.1 GLAIR

La arquitectura GLAIR [Hexmoor, 1993] y [Lammens, 1993] está organizada en tres niveles (figura 36): el nivel de conocimiento (KL), el nivel Perceptuo-Motor (PML), y el nivel Senso-Actuador (SAL). GLAIR integra un sistema simbólico tradicional con un sistema físico en una arquitectura basada en comportamientos. Su característica principal es la presencia de tres niveles distintos con diferentes representaciones y mecanismos de implementación, en particular, la presencia explícita de un nivel de conocimiento. La representación, el razonamiento (incluida la planificación), la percepción y la generación de comportamientos se distribuyen a lo largo de los tres niveles.

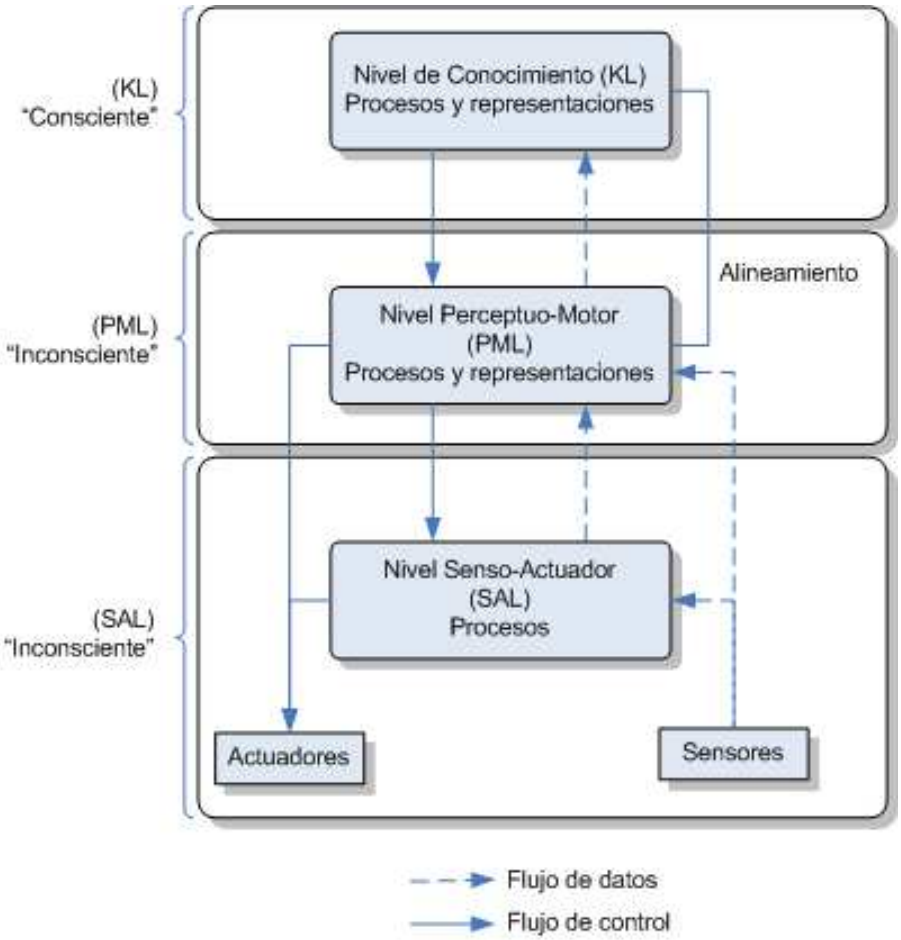

Figura 36. Niveles de la arquitectura de control GLAIR.

El interés se centra en los robots que se encuentran empotrados en un entorno dinámico con el que necesitan interactuar y reaccionar continuamente exhibiendo un comportamiento inteligente. Las características de GLAIR son:

- Se distingue entre el razonamiento consciente y el procesamiento inconsciente de los niveles Perceptuo-Motor y Senso-Actuador.
- Los niveles de la arquitectura son semi-autónomos y se procesan en paralelo.
- El razonamiento consciente tiene lugar a través del razonamiento y la representación explícita de conocimiento.

- El razonamiento consciente guía el comportamiento inconsciente, y los niveles inconscientes, que están continuamente ocupados en el procesamiento de percepción y motor, pueden advertir al nivel consciente de los eventos importantes, tomando el control si fuera necesario. El control y la generación de comportamientos están distribuidos por capas pero no necesariamente de arriba hacia abajo.
- Los mecanismos de bajo nivel pueden expulsar a los de alto nivel. Esto es una forma de inhibición que depende de la capa en la que se encuentren los comportamientos.
- Hay una correspondencia entre los términos en el sistema de razonamiento y representación del conocimiento por una parte, y las sensaciones y percepciones de los objetos, propiedades, eventos y estado del mundo, y capacidades motoras, por otra. A esta correspondencia se le denomina alineamiento.

A continuación se realiza una breve descripción de los niveles que constituyen la arquitectura GLAIR:

- Nivel de conocimiento (KL): el nivel de conocimiento contiene un sistema de planificación y utiliza una representación selectiva de los objetos, eventos y estado del mundo según el curso de acción. El objetivo es modelar sólo las entidades relevantes para la interacción del robot con el mundo. Las representaciones en el nivel de conocimiento son necesarias para razonar sobre las entidades, mientras que las representaciones en el nivel Perceptivo-Motor son necesarias para la interacción física entre entidades.
- Nivel Perceptivo-Motor (PML): el nivel Perceptivo -Motor usa una representación más fina de los objetos, eventos y estados del mundo. En este nivel se debe proporcionar suficiente detalle para hacer posible el control preciso de los actuadores, y los sensores deben ser capaces de suministrar este nivel de detalle para situaciones u objetos determinados. El PML está parcialmente alineado con el KL, en el sentido en que hay una correspondencia entre los identificadores de los objetos en el KL y los objetos del PML. La representación en el PML está englobada con el robot, es decir, depende de la estructura física del robot, sus dimensiones y características particulares. En este nivel se generan varios comportamientos.
- Nivel Senso-Actuador (SAL): el nivel Senso-Actuador es el nivel de las acciones motoras y sensoriales primitivas. En este nivel no hay ninguna representación declarativa explícita de los objetos, sólo hay declaraciones procedimentales (para los actuadores) y datos (en el caso de los sensores). En este nivel se sitúan los “reflejos” que se consideran como bucles de bajo nivel entre los sensores y los actuadores, operando independientemente de los niveles superiores y capaces de expulsar las acciones de éstos.

La división que hace GLAIR en tres niveles, asociando un comportamiento casi “psicológico” a cada nivel es interesante ya que asimila bastante bien la conexión entre las arquitecturas y los sistemas reales que se pretende que éstas implementan.

4.8.2 Sharp

La arquitectura Sharp se ha desarrollado en el INRIA, se describe en [Laugier, 1998], es una arquitectura de tres capas, aunque está orientada a la navegación de un vehículo automóvil a través de una red de carreteras su funcionalidad puede extenderse a más áreas. Sus autores se plantean dos objetivos fundamentales para esta arquitectura:

- Poder tener un control de movimientos eficiente y seguro.
- Obtener un movimiento suave del vehículo, por medio de un adecuado control de la velocidad y la aceleración.

Esta arquitectura adapta las capas deliberativa, secuencial y reactiva de las arquitecturas de tres capas, y las convierte en las denominadas respectivamente, planificador, programador de misiones y controlador de movimientos. Estos componentes se pueden ver en la figura 37.

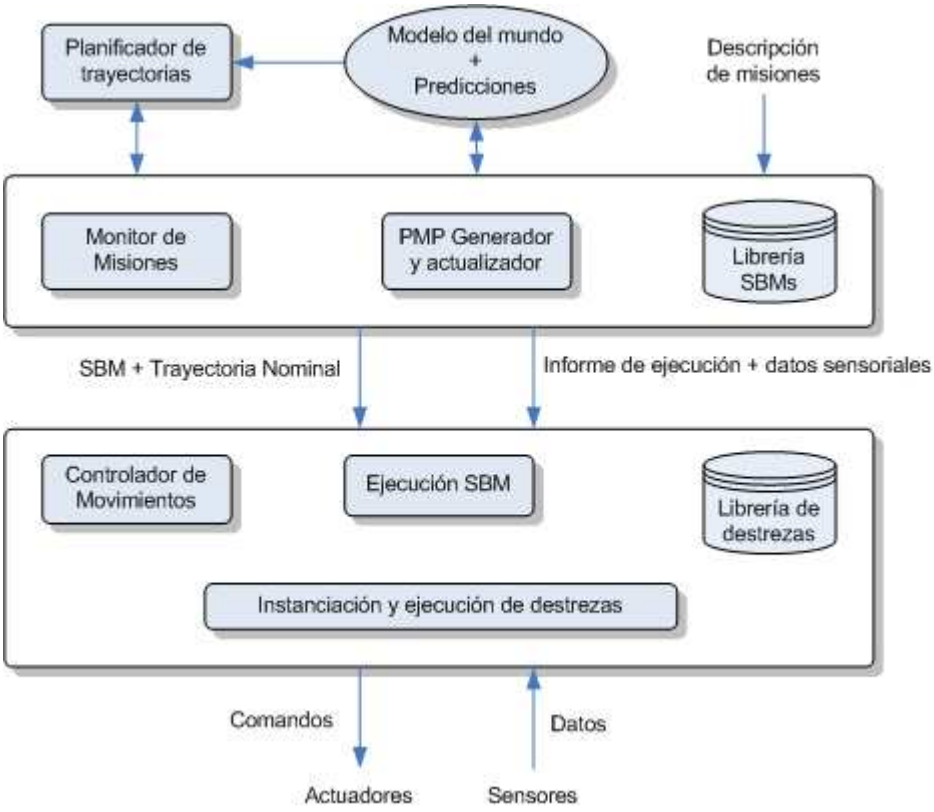

Figura 37. Arquitectura Sharp de control inteligente de la navegación de robots.

Con esta arquitectura se pretende mejorar la eficiencia y robustez de un sistema de navegación autónomo. La mejora se logra por medio de la construcción de planes "on-line". Este concepto se ha llamado "Sensor-Based-Maneuver" (SBM), puede ser visto como una "meta-destreza", se ha elegido este tipo de destreza debido a que, en la navegación en carreteras tiene un conjunto de movimientos finito y no demasiado grande. Esta "meta-destreza" es especialmente útil cuando se trata de replantear secuencias de acciones similares. El concepto de SBM está basado en un concepto de Inteligencia Artificial conocido como "script" (guión). Un guión es una plantilla general que codifica un conocimiento sobre los procedimientos que se deben seguir para realizar alguna tarea específica. Una destreza se empotra dentro de un guión a través de la instanciación de unos parámetros variables en el guión.

La procedencia de estos parámetros puede ser de diversas fuentes como conocimiento previo del sistema, datos de los sensores, o las salidas de otros módulos. Un SBM combina destrezas de control y sensoriales. Las destrezas se consideran funciones elementales con capacidad de trabajar en tiempo real. Las destrezas sensoriales son funciones que procesan en tiempo real los datos de los sensores mientras que las destrezas de control son programas de control (tanto en bucle abierto como cerrado) que generan los comandos adecuados al vehículo. Las destrezas de control pueden usar directamente las salidas de los sensores.

4.8.3 BERRA

La arquitectura BERRA (Behaviour-based Robot Research Architecture) [Lindström, 2000] está desarrollada para un robot de servicio, entendiendo éste como un robot que es capaz de realizar un amplio conjunto de misiones en un entorno clásico de oficina. Para el diseño de la arquitectura se ha tenido en cuenta que una arquitectura que soporte las tareas de un robot de servicio debe proporcionar soporte a:

- Un entorno conceptual reutilizable.
- Una clara distinción entre niveles de competencia.
- Integración simple de nuevos componentes.
- Flexibilidad.
- Rendimiento eficiente en tiempo de ejecución.
- Fácil de depurar.

Aparte de estas características, la arquitectura debe tener ciertas características técnicas que faciliten su desarrollo, tales como el uso de lenguajes estándares o el funcionamiento en una gran diversidad de sistemas operativos. En el caso de la arquitectura BERRA, se ha empleado el lenguaje C++ partiendo de clases abstractas que facilitan la programación de nuevos componentes.

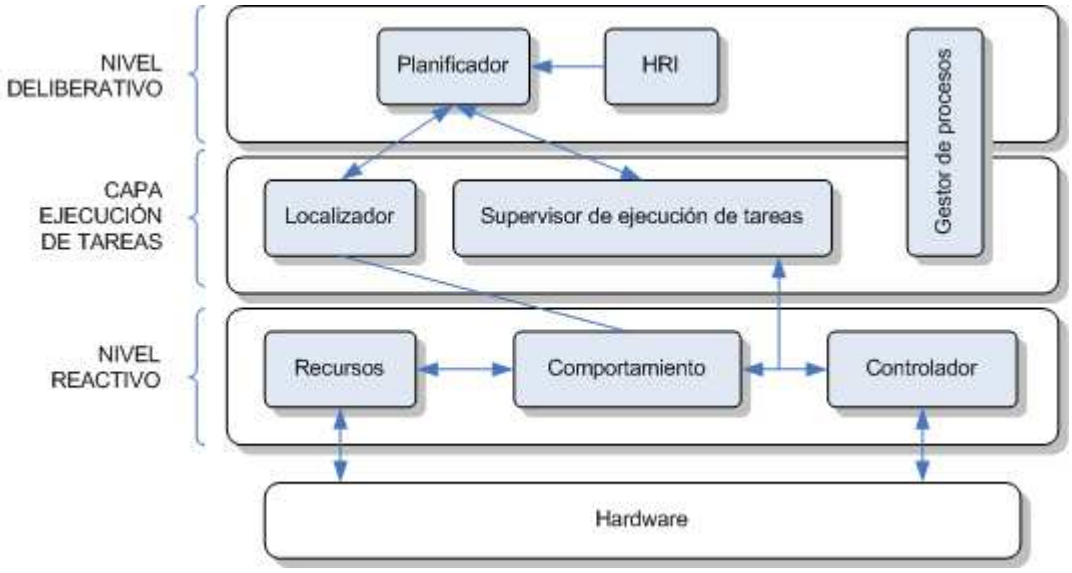

Figura 38. Componentes de la arquitectura de control inteligente BERRA.

La arquitectura es híbrida de tres capas que se pueden observar en la figura 38, con una capa deliberativa que le proporciona la capacidad de realizar los servicios requeridos, la capa reactiva suministra la posibilidad de reaccionar a situaciones inesperadas. En el caso de esta arquitectura la selección de tareas se realiza por medio de una estrategia de planificación y configuración [Arkin, 1998] lo que implica que los módulos de la capa reactiva se pueden configurar y conectar entre ellos por medio de una red flexible.

La capa deliberativa recibe comandos desde un operador humano, estos comandos deben ser entendibles por el sistema. Además, la necesidad de transportar el robot de una localización a otra, hace que este deba tener un conocimiento del entorno que le permita dichos desplazamientos. Estos desplazamientos involucrarán, además, una planificación de la trayectoria. La capa reactiva se basa en comportamientos deberá proporcionar una estrecha conexión entre los sensores y los actuadores. Los sensores se emplean de forma simultánea por varios comportamientos, esto implica la existencia de un mecanismo que comparta la información de los sensores.

El hecho de que las salidas que se deban proporcionar a los actuadores puedan ser variadas, ya que diversos comportamientos pueden requerir diferentes actuaciones por parte de los actuadores, hará que la arquitectura deba soportar un mecanismo de combinación o de fusión de dichos requerimientos. Por último, la capa de ejecución de tareas, funciona como puente que debe salvar el salto existente entre la capa reactiva y la capa deliberativa, permitiendo a esta última monitorizar y configurar la capa reactiva. A continuación se verán con detalle cada uno de los componentes que forman cada capa.

Como componente inicial de la capa deliberativa se tiene el módulo HRI (Human Robot Interface). Este módulo es el enlace entre el operador y el robot, en el caso de las implementaciones, puede ser desde un analizador de voz o de gestos para la entrada y un generador de voz para las salidas de información del robot. La localización de lugares en el entorno, se realiza por medio de los puntos meta.

El otro componente de la capa deliberativa es el planificador, este es el principal componente, y su labor es interpretar y llevar a cabo los comandos que proporciona el operador. El planificador convierte las órdenes en una lista consecutiva de estados y datos de estado. Cada estado representa una cierta configuración de la capa reactiva. Los datos de estados se corresponden con un objetivo, proporcionado en coordenadas globales. Cada estado o datos de estado se transfieren a la capa inferior. Cuando se ha logrado el la tarea encargada, entonces la siguiente tarea se envía a la capa inferior. En caso de llegar un mensaje de error, entonces se revisa el plan, si no se encuentra una solución se le avisa al operador.

En la capa de ejecución de tareas, se encuentra el supervisor de la ejecución de tareas (TES). Tiene como función la gestión de la capa de control reactivo. Este componente recibe un cambio de estado desde el planificador, y lo transfiere a los componentes reactivos.

El localizador es el componente que almacena la información a largo plazo y no es relevante para la capa reactiva. Su función es localizar al robot en el entorno en el que se encuentre. Este componente también se encarga de transformar las coordenadas globales (mundo) a coordenadas locales (odometría). El localizador es quien proporciona a los comportamientos las correcciones sobre la localización del robot o e la odometría del sistema.

Ya en la capa reactiva se encuentran el objeto recurso que proporciona la información del sensor compartiéndola. Un recurso es un servidor cuyos clientes son los comportamientos, sin embargo, un recurso puede ser un cliente de otros recursos para poder implementar un nivel de abstracción alto. Cuando un recurso no tiene clientes, pasa automáticamente al estado de ocioso. Para que un cliente reciba la información requerida por parte de un recurso, debe establecer una conexión y solicitad un esquema de suscripción, los tipos existentes son:

- El cliente envía peticiones cada vez que el solicita un nuevo dato (paradigma pull)
- El cliente debe ser actualizado cada vez que un nuevo dato está disponible en el sensor (paradigma push).
- El cliente especifica el intervalo de tiempo entre actualizaciones (paradigma asíncrono)

El comportamiento del sistema de comunicaciones es bastante dependiente del paradigma que se emplee y de la cómo se reparte la cantidad de información enviada (toda unida o diversos fragmentos), así como si se envía la información directamente o como referencia.

La conexión entre sensores y actuadores la realizan los comportamientos. Un comportamiento puede conectar uno o más recursos y permite conexiones realizadas por el controlador (componente descrito más adelante) los comportamientos pueden ser activados por medio de intervalos de tiempo o dependiendo de los datos que circulen por el canal de comunicaciones. Pueden aceptar los datos de control desde la capa de ejecución de tareas.

Los controladores son los responsables de enviar las órdenes a los actuadores del robot. Ante un cambio de estado, el controlador recibe nuevas órdenes por parte de la capa de ejecución de tareas (concretamente del TES), ante estas órdenes el controlador conecta los comportamientos correspondientes. Los datos recibidos de los sensores son fusionados para producir las señales de control correspondientes.

4.8.4 SSS

SSS (Servo, Subsumption, Symbolic) es una arquitectura de tres capas, combinando una capa de servo-control, una capa basada en subsumption y una capa simbólica [Connell, 1992]. El objetivo de la arquitectura es combinar las mejores características de un sistema convencional de servo-control con sistemas multi-agentes y sistemas de inteligencia artificial.

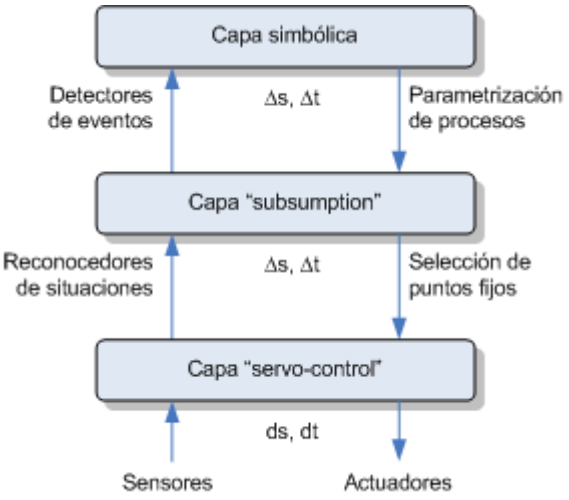

Figura 39. Capas de la arquitectura SSS de control inteligente.

Las tres capas de la arquitectura, van progresivamente cuantificando, primero el espacio (s en la figura 39) y luego el tiempo (t en la figura 39). La capa de servo-control, prácticamente opera en el dominio continuo del tiempo y el dominio continuo del espacio. Es decir, estos sistemas, constantemente monitorizan el estado del mundo y típicamente representan este espacio como un conjunto de valores escalares.

La capa que se basa en un sistema basado en comportamientos (subsumtion), también chequea constantemente sus entradas sensoriales, pero sus representaciones tienden a ser situaciones concretas, proporcionadas por la capa inferior. Por esto, la capa de subsumtion discretiza los posibles estados del mundo en un pequeño número de categorías especiales dependientes de tareas.

El sistema simbólico da un paso más y discretiza también el tiempo en base a eventos significativos. Normalmente se emplean términos del tipo “después de X hacer Y” o “hacer A antes de que ocurra B”. Debido a que los eventos temporales se discretizan a partir de los cambios en situaciones espaciales, es decir hace falta una discretización espacial para que se produzca una discretización temporal, no hay un bloque o una capa en la que haya una discretización temporal previa a la discretización espacial.

Para emplear y poder obtener el máximo partido a las tres capas, se hace necesario definir unos interfaces entre las capas que tengan una alta eficiencia. La primera interfaz es la transformación en comandos u órdenes entre la capa de comportamientos y los servos que controlan al robot. La transformación se hace, básicamente, teniendo en cuenta que para una tarea concreta, ciertos tipos de estado de los sensores son equivalentes y llamarán a la misma respuesta por parte del motor. El interfaz entre la capa simbólica y la capa “subsumtion”, consiste en la capacidad de activar o desactivar selectivamente cada comportamiento. Los eventos de activación (o desactivación) de los comportamientos permanecen activados o desactivados mientras la capa simbólica no decida lo contrario. El interfaz entre la capa basada en comportamientos y la capa simbólica se logra por un mecanismo que controla si las situaciones por las que va pasando el robot son correctas. Por ejemplo, si el robot no ha alcanzado su destino, e informa de que no ha realizado progresos recientemente, se genera un evento de “camino bloqueado”. Para desacoplar el sistema simbólico del sistema en tiempo real, se dispone de una “tabla de contingencias”, esta tabla permite al sistema simbólico pre-compilar qué acciones se toman cuando cierto evento se da. Las entradas de esta tabla, reflejan lo que la capa simbólica espera que ocurra.

4.8.5 Payton's Architecture

Esta arquitectura propuesta en [Payton, 1986] está basado en una descomposición jerárquica en la que cada capa se caracteriza por un tipo de procesamiento de datos de los sensores. Esta arquitectura se compone de un sistema de percepción estructurado en capas con cuatro módulos principales (figura 40).

El planificador de misiones define una secuencia de objetivos geográficos para alcanzar a teniendo en cuenta las restricciones de movimiento. El planificador basado en mapa usa un modelo del mundo global para generar caminos conectando los objetivos geográficos previamente definidos en la capa superior (el tiempo de respuesta de este planificador puede durar una gran cantidad de minutos. El planificador local de caminos determina los detalles de los movimientos que son necesarios para mover al robot a lo largo del camino planificado (el tiempo de respuesta de este planificador es de unos pocos segundos). El planificador reflexivo controla en tiempo real la ejecución de las tareas de movimiento.

Figura 40. Arquitectura Payton's de control de la navegación de robots.

Desde el punto de vista de la implementación esta arquitectura se ha desarrollado usando agentes expertos comunicados por medio de una pizarra distribuida. Usando esta aproximación, la actividad de un módulo particular puede, teóricamente, ser controlado por un una capa superior por medio de la selección de agentes expertos que activan los módulos. Sin embargo, sólo se ha descrito la implementación de algunos agentes expertos que se sitúan en el planificador reflexivo (por ejemplo seguir un muro, o evitar un obstáculo y otros). En esta implementación, los comportamientos reflexivos se asocian con algunos sensores virtuales con el objetivo de proveer información especializada (por ejemplo la detección de obstáculos, reconocimiento de objetos, localización en el entorno y otros). La activación de los comportamientos reflexivos más apropiados para cada situación se da por medio de la comunicación prioritaria por medio de la pizarra distribuida.

Con esta aproximación se han planificado y ejecutado completamente diversas misiones complejas con un nivel de reactividad significante. Las principales limitaciones del sistema provienen por las limitaciones del mecanismo de comunicación entre las distintas capas y las combinaciones de comportamientos.

4.9 Análisis

4.9.1 De las arquitecturas en general

En este apartado se ha realizado una revisión de diversas arquitecturas para la navegación de robots móviles, y es fácil apreciar la diversidad de modelos existentes. La pluralidad de modelos y referentes da lugar a pensar en un campo de investigación abierto y en continua evolución, donde encontrar un solución integral y unificada es casi utópico. Habitualmente una arquitectura se desarrolla para un modelo de robot más o menos concreto, que debe realizar sus actividades en una serie de entornos específicos. Si se tiene en cuenta que la mayoría de las arquitecturas están inspiradas en modelos naturales, el hecho de que exista una variedad tan amplia de arquitecturas no es nada diferente de lo que ocurre en la naturaleza, donde también hay una gran variedad de especies dependientes de las funciones y entornos en los que habitan.

Sin embargo, en la naturaleza sí que existen unos aspectos comunes a todos los seres vivos. Todos los seres vivos reciben estímulos tanto externos como internos, y los procesan para dar lugar a acciones tanto internas como externas. Los estímulos proporcionan la información con la que los seres pueden desarrollar su actividad. El mismo estímulo puede desencadenar diferentes procesos y acciones incluso en el mismo ser. En un ser humano, la llama de una vela puede producir un movimiento rápido de alejamiento del brazo si la llama está muy cerca de algún sensor de calor de la piel; esa misma llama puede desencadenar el proceso contrario que mueva el brazo hacia la vela para cogerla y alumbrar algún rincón oscuro. La diferencia está tanto en el nivel al que llega la información sensorial como el procesamiento que se hace de la misma, antes de que llegue al nivel, o en el mismo nivel. La arquitectura interna del ser será la que determine qué tipo de actividad será la que podrá desarrollar. De la misma manera, en todas las arquitecturas de control y navegación de robots aparecen componentes o bloques que se pueden organizar en niveles.

En la navegación controlada por el nivel reactivo, el robot debe reaccionar rápidamente ante los cambios del entorno. Por este motivo, los requerimientos de información de los algoritmos son simples, lo cual no es problemático. Además las fuentes de datos (generalmente los sensores) están muy cerca de los agentes que deben procesar la información. En el nivel reactivo, o los niveles cercanos a éste, los requerimientos del sistema de comunicaciones son principalmente: tiempo real en las transmisiones y fiabilidad. Sin embargo, en el nivel deliberativo, o a medida que los niveles se acercan más al nivel deliberativo, el robot necesita reaccionar inteligentemente ante el entorno. Es por esto, que en el nivel deliberativo, se requiere una información más elaborada y compleja. Las fuentes de datos (sensores internos o externos de otros robots) están más distantes, tanto en el tiempo como en el espacio, de lo que lo estaban de los niveles reactivos. Los requerimientos del sistema de comunicación en la capa deliberativa, básicamente son: control de tiempos, sin llegar a ser tan crítico como el tiempo real, fiabilidad, flexibilidad, y adaptabilidad. La diversidad de agentes usados en la navegación deliberativa, debido a la diversidad de comportamientos inteligentes que requiere el nivel, implica que los datos de la misma fuente (por ejemplo del mismo sensor) pueden ser usados de formas distintas dependiendo de los requerimientos de los agentes. Esta diversidad, significa una variedad de procesado de los datos a medida que se acercan a la capa deliberativa. El sistema de comunicaciones puede colaborar con las capas, a medida que se incrementa su función deliberativa, distribuyendo, organizando, filtrando añadiendo integridad o preprocesando la información que transporta.

4.9.2 Idoneidad para la distribución

Destacar algunas arquitecturas sobre otras sería, además de injusto, un error ya que todas las arquitecturas son válidas en los entornos para los que han sido desarrollados. De todas las arquitecturas anteriores se podrían destacar muchas características, entre ellas se han seleccionado algunas, especialmente aquellas que no tienen en la especificidad de los componentes sus características.

El nivel de modularidad de cada arquitectura se ha entendido como lo divisible que puede ser el control y por tanto lo adaptado que está a una situación concreta con pequeños cambios. Asimismo también se ha valorado la modularidad de la capa deliberativa, es decir, cómo los módulos de ésta capa están diseñados en función de unas condiciones de trabajo concretas de la arquitectura. La modularidad podrá ser un buen indicador de la **capacidad de distribución** que tiene una arquitectura y consecuentemente lo **escalable** que ésta resulta.

Tabla 3. Valoración de la modularidad de las arquitecturas revisadas.

Arquitectura	Tipo	Subtipo	Modularidad Reactiva	Modularidad Deliberativa	Modularidad General
Subsunción	Reactiva	Jerárquica	Alta	Alta	Muy alta
JPL	Deliberativa	Secuencial	Baja	Baja	Muy baja
NASREM	Deliberativa	Paralelo	Alta	Alta	Muy alta
AuRA	Híbrida	Organizativa	Alta	Baja	Media
SFX	Híbrida	Organizativa	Alta	Baja	Media
LAAS	Híbrida	Organizativa	Alta	Baja	Media
3T	Híbrida	Jerárquica de estados	Alta	Alta	Muy alta
ATLANTIS	Híbrida	Jerárquica de estados	Media	Alta	Alta
Saphira	Híbrida	Orientada a modelos	Media	Baja	Baja
TCA	Híbrida	Orientada a modelos	Media	Baja	Baja
GLAIR	Híbrida	Niveles	Baja	Media	Baja
Sharp	Híbrida	Niveles	Media	Media	Media
BERRA	Híbrida	Niveles	Baja	Media	Baja
SSS	Híbrida	Niveles	Media	Media	Media
Payton’s	Híbrida	Niveles	Media	Media	Media

Como se puede deducir de la valoración anterior, una de las características destacables para que la arquitectura sea adaptable es la posibilidad de generar comportamientos avanzados sin necesidad de un alto coste organizativo, como sucede en las arquitecturas deliberativas. Sin embargo, la organización de los componentes, especialmente la posibilidad de generalizar sus funciones es una de las funcionalidades que tienen las arquitecturas adaptables. A continuación se procederá a intentar extraer los aspectos particulares de algunas de las arquitecturas anteriores que tienen una valoración media o alta tanto en la capa reactiva como en la deliberativa.

Posiblemente la arquitectura más nombrada en las publicaciones de control, es la de subsunción [Brooks, 1986], no en vano es una de las que implementa de forma más **homogénea y elegante** un control deliberativo como **jerarquización basada en niveles** priorizados. Los comportamientos de supervivencia básicos son los más prioritarios, mientras que los menos prioritarios son los puramente deliberativos basados en el razonamiento. Sin embargo, en éste tipo de arquitecturas unas condiciones de un entorno continuamente cambiante, por ejemplo, pueden hacer que los comportamientos deliberativos no terminen emergiendo debido a la prioridad de los comportamientos reactivos.

De la arquitectura NASREM [Albus and Barbera, 2005] cabe destacar algunos aspectos. La **estructuración homogénea** independientemente del nivel, reactivo o deliberativo, en el que se opere, la hace muy adaptable. Además el emplear un método para **almacenar y compartir el conocimiento global** la hace muy conveniente para el aprendizaje.

La **reutilización de componentes diferenciados para el control reactivo** es un patrón común en varias de las arquitecturas. En la arquitectura AuRA [Arkin, 1990] la división del control en módulos se realiza por medio del empleo de **esquemas reutilizables** en la capa reactiva hace que sea muy adaptable a diversos entornos. Algo similar sucede en SFX [Murphy, 2000] donde los esquemas son sustituidos por **comportamientos comunicados con agentes**. En LAAS [Alami et al., 2000] la división se realiza directamente en los denominados módulos de control. En la arquitectura 3T [Bonasso, 1997], esa división se realiza en las denominadas **habilidades**, accionadas por eventos externos al sistema y planificada de una forma jerárquica.

En cuanto al control deliberativo, en la mayor parte de las arquitecturas es realizado por módulos muy específicos para cada uso. Sin embargo, algunas de las arquitecturas buscan una modularidad interesante, lo que permitiría mejorar tanto el diseño de los mismos como la distribución en caso de tener que funcionar en un sistema distribuido.

En subsunción el comportamiento cercano al deliberativo se presupone que aparece como comportamiento emergente a partir de los comportamientos de los módulos reactivos. Posiblemente el mejor ejemplo de modularidad se localiza en NASREM donde los módulos deliberativos son tratados de la misma forma que los reactivos, lo que hace que ésta arquitectura tenga una homogeneidad muy elegante. En la arquitectura 3T la modularidad deliberativa se logra por medio de una jerarquía de objetivos cuya sucesiva transformación en subobjetivos conlleva la asignación automática de tareas. En ATLANTIS los módulos son homogéneos por medio de procesos LISP.

5 Conclusiones

5.1 Sobre las arquitecturas

De la revisión realizada a las arquitecturas domoticas, se puede observar cómo éstas están más adaptadas a la distribución de sus componentes que las arquitecturas de navegación. Esto se debe fundamentalmente a la aparición posterior de éstas y el consecuente aprovechamiento de las tecnologías de comunicaciones. Este aspecto es interesante, ya que implica que el diseño de una arquitectura se debe basar en varios aspectos. Además de la funcionalidad a la que está destinada la arquitectura, se debe tener en cuenta la topología del control, lo que implicará **tener en cuenta las comunicaciones como parte de la arquitectura** y por tanto del diseño de la misma.

La modularidad que aparece en prácticamente todas las arquitecturas domóticas no se da tanto en las arquitecturas de control de la navegación de robots. Este aspecto puede deberse a la funcionalidad más sencilla que tienen los sistemas domóticos actuales, centrados en un confort y con escasa implantación de aspectos inteligentes como el aprendizaje o la adaptación. Es de esperar que, a medida que vayan evolucionando los sistemas domóticos, la concreción de sus módulos vaya en aumento, y consiguientemente la similitud con los niveles deliberativos de las arquitecturas de navegación de robots.

En las arquitecturas domóticas, la tendencia es a disponer de una infraestructura que permita un sistema distribuido, heterogéneo, con escalabilidad. Esto se logra por medio de los sistemas basados en servicios. Para ello los sistemas deben proporcionar un sistema de descubrimiento de dispositivos.

En las arquitecturas de navegación de robots, El paradigma híbrido es el más adecuado para mantener una navegación que contenga las ventajas del paradigma reactivo, con el cumplimiento de las de restricciones temporales y las ventajas del paradigma deliberativo con la capacidad de realización de misiones complejas

Aprovechar la experiencia adquirida en tantos años de diseño de sistemas de navegación de robots para afrontar el diseño de las arquitecturas domóticas, y porqué no las industriales es uno de los retos más interesantes en el aspecto de las arquitecturas de control distribuido.

5.2 Sobre las características

A partir de las revisiones de las características deseables de las arquitecturas tanto de sistemas domóticos como de sistemas de navegación de robots se han organizado en siguiente tabla.

Cabe destacar que algunas de las características pueden ser comunes, pero están denominadas de diferentes formas, dependiendo del autor. A todo ello se debe tener en cuenta que algunas características, como la versatilidad, son más genéricas que otras, como la programabilidad.

Tabla 4. Características de los sistemas de control distribuido.

Característica	[Aiello, 2005]	[Brooks, 1991]	[Arkin, 1998]	[Alami, 1998]	Relevancia
Apertura	Sí				Baja
Escalabilidad	Sí	Sí		Sí	Alta
Heterogeneidad	Sí				Baja
Topología	Sí				Baja
Generalidad		Sí			Baja
Versatilidad		Sí			Baja
Adaptabilidad		Sí	Sí	Sí	Alta
Racionalidad		Sí			Baja
Aprendizaje		Sí			Baja
Planificación		Sí			Baja
Reactividad		Sí		Sí	Media
Eficiencia		Sí			Baja
Modularidad			Sí		Baja
Portabilidad			Sí		Baja
Robustez			Sí	Sí	Media
Programabilidad				Sí	Baja
Coherencia				Sí	Baja

Determinar las idóneas de una arquitectura es una tarea complicada, ya que algunas están enfrentadas, como la escalabilidad y la heterogeneidad. Conjuntar todas es un reto muy interesante.

5.3 Optimización de sistemas de control distribuido

La principal característica de los sistemas de control distribuido es que no trabajan con sistemas controlados ni predecibles, por lo que deben ser sistemas robustos ante condiciones inesperadas y se deben adaptar a los posibles cambios y a los fallos de los subsistemas. Dado que el objetivo es la optimización del rendimiento del sistema, es interesante averiguar las características necesarias y así obtener los principales requerimientos de los sistemas ciber físicos. Un esquema conceptual de éstas características se puede observar en la siguiente figura.

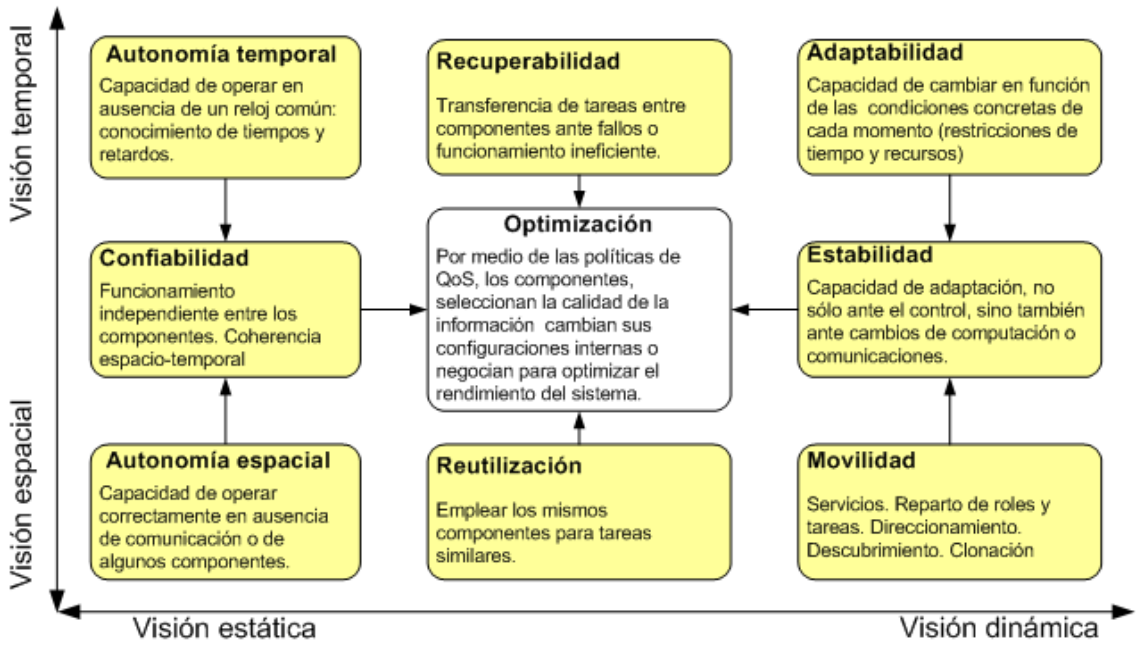

Figura 41. Principales características de los sistemas de control distribuido y sus relaciones.

A continuación se explican detalladamente cada una de las características expuestas en la figura anterior.

- Autonomía temporal. La autonomía temporal de los componentes del sistema consiste en la capacidad que tienen éstos de poder prescindir de un reloj común, y sin embargo poder obtener información temporal y ofrecerla al usuario para cumplir requisitos temporales.
- Autonomía espacial. La autonomía espacial se refiere a la posibilidad que tienen los componentes de poder continuar trabajando independiente de parte de la información que requieren para su funcionamiento. A mayor independencia espacial, se podrá minimizar en mayor medida el impacto de ésta ausencia en el rendimiento del sistema.
- Confiabilidad. La confiabilidad es una característica derivada de las dos anteriores. Un sistema que proporciona un control temporal y tenga una cierta autonomía del resto de los componentes es un sistema en el que se puede confiar el control. La confiabilidad es una extensión del concepto clásico de fiabilidad, que habitualmente se delimita a la ausencia de errores.
- Recuperabilidad. La recuperabilidad trata de reorganizar el sistema ante cambios de condiciones de funcionamiento. Es un concepto que implica del sistema la capacidad de transferencia de información entre todos los componentes independientemente de sus características, por lo se hace necesario el empleo de meta datos en el proceso.
- Reutilización. Esta característica se refiere a la posibilidad que el sistema debe ofrecer de poder emplear el mismo componente para situaciones distintas. Es un concepto estático que parte del concepto de reutilización de código pero extendido a los componentes del sistema.
- Adaptabilidad. Es una de las características clave, por no decir la más importante cara a lograr una optimización del sistema. La adaptabilidad es la capacidad de

proporcionar una respuesta robusta del sistema ante un cambio en las condiciones del mismo. Al tratarse de sistemas ciber físicos, los cambios de condiciones se refiere tanto a las condiciones de realidad física sobre la que el sistema trabaja, como las condiciones de comunicaciones o de cómputo.

- Estabilidad. La estabilidad se entiende como la capacidad de mantenerse dentro de unos márgenes de funcionamiento, no sólo ante el control, sino también ante cambios de computación o comunicaciones.
- Movilidad. La movilidad es la capacidad de poder reubicar espacialmente los componentes. De esta forma se puede realizar un reparto de roles y tareas en el sistema. Para lograr la movilidad de componentes o las partes del mismo, se debe proporcionar un direccionamiento común del sistema que abstraiga los detalles a los componentes. Además los componentes deben tener la capacidad de descubrir las ubicaciones donde poder trabajar.

Lograr estas características no es gratuito, a medida que se van aumentando las prestaciones que facilitan optimizar un sistema aparecen algunas dificultades. Por ejemplo, la movilidad de componentes permite clonar los mismos para aumentar la fiabilidad de un sistema, sin embargo, esa clonación implica un aumento en la redundancia de la información que se proporciona por parte de los componentes.

5.4 Posibles áreas de investigación en arquitecturas de control distribuido

A partir de lo expuesto anteriormente, son muchas las líneas de investigación que se pueden poner en marcha. A medida que la variedad de sistemas de control van pasando de modelos centralizados a modelos distribuidos, la importancia de las comunicaciones es mayor, por ello estudiar la relación de las comunicaciones y el control es un aspecto muy interesante.

El diseño de los sistemas de control distribuido, teniendo en cuenta la adaptabilidad que el sistema debe tener es otra de las áreas de interés. Poco a poco las arquitecturas han evolucionado hacia una modularidad que inicialmente no tenían. El soporte de la arquitectura a la modularidad poco a poco será importante. Para ello, el control deberá basarse en componentes conectados, en cierta medida, entre sí.

La medición de la eficiencia en las arquitecturas es otra de las características necesarias. Inicialmente la eficiencia estaba basada exclusivamente en parámetros temporales sobre los objetivos cumplidos en las misiones. El dinamismo implícito de los sistemas actuales hace que otros parámetros, como el aprendizaje o la adaptación vayan teniendo más relevancia.

En la Figura 42, se puede observar cómo los sistemas domóticos van evolucionando hacia sistemas de control distribuido similares a los sistemas de navegación de robots, especialmente en el aspecto deliberativo, inicialmente la domótica sólo cubría aspectos tan sencillos como el control de luces o de escenarios de confort. Sin embargo, poco a poco los sistemas domóticos van requiriendo más inteligencia, para el aprendizaje de comportamientos o la gestión energética y de más restricciones temporales, especialmente en los aspectos de seguridad. El aprovechamiento de los conocimientos en arquitecturas de robots puede ser aprovechado perfectamente para los avances en domótica.

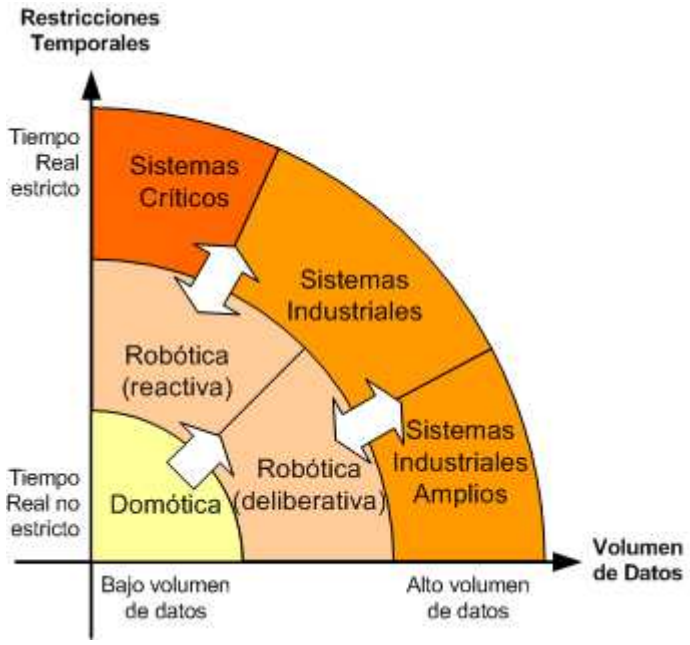

Figura 42. Evolución de los sistemas de control distribuido.

De manera análoga a la presentada en la Figura 42 para los sistemas domóticos. Los sistemas industriales también se ven afectados por los avances obtenidos en los sistemas similares de robótica, e incluso en los avances en domótica. El buen diseño de las arquitecturas domóticas, puede incidir a largo plazo en el control de sistemas industriales distribuidos.

La gestión de un bajo volumen de datos con escasas restricciones temporales, como es el caso de los sistemas domóticos, encaja a la perfección con el paradigma del control basado en eventos. Los avances en estos aspectos pueden aplicarse a las arquitecturas de control de la navegación de robots.

6 Referencias

Abhishek et al., 2003 Abhishek Roy, Soumya K. Das Bhaumik, Amiya Bhattacharya, Kalyan Basu, Diane J. Cook, and Sajal K. Das. Location aware resource management in smart homes. In First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), 2003.

Aiello, 2005 Aiello, Marco (2005) The Role of Web Services at Home. Technical Report DIT-05-065, Informatica e Telecomunicazioni, University of Trento.

Alami et al., 1998 R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand (1998). An architecture for Autonomy. International Journal of Robotics Research, Vol. 17, No. 4, pp. 315-337.

Alami et al., 2000 R. Alami, R. Chatila, S. Fleury, M. Herrb, F. Ingrand, M. Khatib, B. Morisset, P. Montarlier, and T. Simeon, (2000). Around the lab in 40 labs... In IEEE International Conference on Robotics and Automation, ICRA'O0, San Francisco.

Albus and Barbera, 2005 J.S. Albus and A.J. Barbera, "RCS: A cognitive architecture for intelligent multi-agent systems," In Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (2004).

Albus, 1991 Albus, J.S., (1991). "A Theory of Intelligent Machine Systems". IEE/RSJ International Workshop on Intelligent Robots and Systems. Osaka, Japan.

Albus, 1993 Albus, J.S.(1993) A Reference Model Architecture for Intelligent Systems Design. In Antsaklis, P.J., Passino, K.M. (Eds.) (1993) An Introduction to Intelligent and Autonomous Control. Kluwer Academic Publishers, 1993, Chapter 2, pp27-56. ISBN 0-7923-9267-1

Arkin, 1989 Arkin, R.C., (1989). Motor schema-based mobile robot navigation. International Journal of Robotics Research, Vol. 8, No. 4, pp. 92-112.

Arkin, 1990 Arkin, R.C., (1990). Integrating Behavioural, Perceptual and World Knowledge in Reactive Navigation. Robots and Autonomous Systems, 6, 105-122.

Arkin, 1998 Arkin, R.C., (1998). Behavior-Based Robotics. MIT Press, Cambridge.

Bizzarri, 1999 Bizzarri, Maurice (Nov. 1999). Home API: A network-independent home control architecture. The Universal Plug and Play Forum.

Bonasso, 1995a Bonasso, R.P., Kortenkamp, D., Miller, D.P., and Slack, M., (1995), Experiences with an architecture for intelligent reactive agents, Internal Report Metrica Robotics and Automation Group, NASA Johnson Space Center.

Bonasso, 1995b R. P. Bonasso, D. Kortenkamp, D. Miller and M. Slack Experiences with an Architecture for Intelligent, Reactive Agents In Intelligent Agents II: Agent Theories, Architectures, and Languages, ed. Michael Wooldridge, Joerg P. Mueller, and Milind Tambe, Springer-Verlag, 1995

Bonasso, 1997 Bonasso, R.P., Firby, J., Gat, E., Kortenkamp, D., Miller, D., Slack, M., (1997). A Proven Three-tiered Architecture for Programming Autonomous Robots. Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, no. 2.

Braitenberg, 1984 Braitenberg, V., (1984). Vehicles: Experiments in Synthetic Psychology. MIT Press, Cambridge.

Brooks, 1986 Brooks, R.A., (1986). A robust layered control architecture for a mobile robot. IEEE Journal of Robotics and Automation, 2, (1), 14-23.

Brooks, 1991a Brooks, R.A., (1991) "Integrated Systems Based on Behaviors", SIGART Bulletin (2:4), August 1991, pp. 46–50.

Brooks, 1991b Brooks, R.A., (1991). Intelligence without representation. Artificial Intelligence, 47, pp. 139-159.

Chao-Lin et al., 2004 Chao-Lin Wu, Wei-Chen Wang, Li-Chen Fu. Mobile agent based integrated control architecture for home automation system. Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, Vol.4, Iss., 28 Sept.-2 Oct. 2004. Pages: 3668- 3673 vol.4

Cheng-Fa and Hang-Chang, 2002 Cheng-Fa Tsai and Hang-Chang Wu, “Massihn: A Multi-Agent Architecture for Intelligent Home Network System,” IEEE Transactions on Consumer Electronics Vol.48, Issue 3, pp.505-514, August 2002.

Chown, 1999 Chown, E., (1999). Making predictions in an uncertain world: Environmental structure and cognitive maps. Adaptative Behaviour 7 (1), 17-33.

Connell, 1992 Connell, J.; SSS: A Hybrid Architecture Applied to Robot Navigation; IEEE International Conference on Robotic and Automation, IEEE Press, California (1992)

Cook and Das, 2007 Diane J. Cook , Sajal K. Das, How smart are our environments? An updated look at the state of the art, Pervasive and Mobile Computing, v.3 n.2, p.53-73, March, 2007.

Cook et al., 2003 D. J. Cook, M. Youngblood, E. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, MavHome: An Agent-Based Smart Home. In proceeding of the Conference on Pervasive Computing, 2003.

Coste-Manière, 2000 Coste-Manière, E. & Simmons, R., (2000), “Architecture, the Backbone of Robotic Systems”, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA.

De Carolis and Cozzolongo, 2004 Berardina De Carolis, Giovanni Cozzolongo: C@sa: Intelligent Home Control and Simulation. International Conference on Computational Intelligence 2004: 462-465

Gat, 1990 Gat, E., M. Slack, D.P. Miller and R.J. firby. 1990. Path Planning and Execution Monitoring for a Planetary rover. In Proc. of the IEEE Int. Conf. on Robotics and Automation. Vol. 1 Cincinatti, OH (US): pp. 20-25.

Gat, 1991 Gat, E., (1991). "Reliable Goal-directed Reactive Control for Real-world Autonomous Mobile Robots". Ph.D. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Gat, 1992 Gat, E., (1992). Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real world mobile robots In Proceedings of the National Conference on Artificial Intelligence. San Jose (CA): pp. 809-815.

Hexmoor, 1993 H. Hexmoor, J. Lammens, and S. C. Shapiro. Embodiment in GLAIR: A grounded layered architecture with integrated reasoning for autonomous agents. In D. Dankel, editor, Proceedings of the Florida AI Research Symposium, pages 325--329, 1993.

IEEE, 2000 IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, Sponsor: Software Engineering Standards Committee of the IEEE Computer Society, Approved 21 September 2000. (ISO/IEC 42010:2007)

Konolige, 1998 Konolige, K., Myers, K., (1998). The Saphira Architecture for Autonomous Mobile Robots. Artificial Intelligence and Mobile Robots. D. Kortenkamp, R. Bonasson, R. Murphy, editors, MIT Press, 1998.

Lammens, 1993 Lammens, J. M.; Hexmoor, H. H.; and Shapiro, S. C. 1995. Of elephants and men. In Steels, L., ed., The Biology and Technology of Intelligent Autonomous Agents. Berlin: Springer-Verlag, Berlin. 312--344

Laugier, 1999 C.Laugier, Th. Fraichard, ph. Garnier, I. E. Paromtchik, and A. Scheuer. Sensor-Based control architecture for a car-like vehicle. Autonomous Robots, 6(2), May 1999

Lindström, 2000 Lindström, M., Orebäck, A., and Christensen, H. 2000. Berra: A research architecture for service robots. In Internacional Conference on Robotics and Automation.

Miori et al., 2006 Miori, V.; Tarrini, L.; Manca, M.; Tolomei, G. An open standard solution for domotic interoperability. Consumer Electronics, IEEE Transactions on, Vol.52, Iss.1, Feb. 2006. Pages: 97- 103.

Mozer, 1998 M. C. Mozer. The neural network house: An environment that adapts to its inhabitants. In Proceedings of the AAAI 1998.

Mozer, 2004 — Mozer, M.C., Lessons from an adaptive home. In: Cook, D.J., Das, S.K. (Eds.), Smart Environments: Technology, Protocols, and Applications, Wiley. pp. 273-298. 2004.

Murphy and Arkin, 1992 — R. R. Murphy, R. C. Arkin, "Sfx: An Architecture For Action-oriented Sensor Fusion", Proceedings of IEEE/RSJ Intelligent Robots and Systems, Vol. 2, July 7-10, pp: 079-1086, 1992.

Murphy, 2000 — Murphy, R.R., (2000). Introduction to AI Robotics. MIT Press.

Nii, 1989 — 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.

Nilsson, 1980 — Nilsson, N.J., (1980). Principles of Artificial Intelligence. Morgan Kaufmann, Ed. Los Altos, C.A.

Orebäck and Christensen, 2003 — Orebäck, A. y Christensen, H.I., (2003). Evaluation of Architectures for Mobile Robotics. Autonomous Robots 14, pp. 33-49.Kluwer Academic Publishers.

Payton, 1986 — Payton, D.W. 1986. An Architecture for Reflexive Autonomous Vehicle Control. In Proc. of the IEEE Int. conf. on Robotics and Automation. San Francisco, CA (US): pp. 1983-1845

Petriu et al., 2000 — E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, V.Z. Groza, Sensor-based information appliances, IEEE Instrumentation and Measurement Magazine 3 (4) (2000) 31–35.

Rosenblatt and Payton, 1989 — Rosenblatt, J. y Payton, D. A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control. International Joint Conference of Neural Networks. Washington D.C. II-317, June 1989.

Ruyter et al., 2005 — Ruyter, B.E.R. de, Aarts, E.H.L., Markopoulos, P., IJsselsteijn, W.A. (2005). Ambient intelligence research in homelab: Engineering the user experience In W. Weber, J.M. Rabaey, E. Aarts, Ambient Intelligence (pp. 49-62) Berlin: Springer Verlag.

Simmons, 1994 — R. Simmons. Structured control for autonomous robots. IEEE Transactions on Robotics and Automation, 10(1):34-43, Feb. 1994.

Wang et al., 2000 — Yi-Min Wang, Wilf Russell, Anish Arora, Rajesh Jagannathan, Jun Xu. Towards Dependable Home Networking: An Experience Report. Dependable Systems and Networks. 2000.

7 ANEXO: Relación de las arquitecturas analizadas

Nombre	Significado	Referencia principal
3T	3 Tiered	Bonasso, 1996
ACT-R		Anderson & Lebiere, 1998
AFREB	Adaptative Fusion of REactive Behaviours	Moreno et al., 1996
AIS	Adaptative Intelligent Systems	Hayes-Roth, 1995
ALLIANCE		Parker, 1998
ARTIS		García-Fornés, 1995
ATLANTIS	A Three-Layer Architecture for Navigating Through Intrincate Situations	Gat, 1992
AuRA	Autonomous Robot Architecture	Arkin, 1989
BERRA	Behaviour-based Robot Research Architecture	Lindström, 2000
BOID	Belief, Obligations, Desire, Intention	Broersen, 2001
CELLO		Occello, 1998
DAMN	Distributed Architecture For Mobile Navigation	Rosenblatt, 1995
ERE	Entropy Reduction Engine	Drummond, 1991
FSA	Frame Sensor-Adapter	Posadas, 2007
GLAIR	Grounded Layered Architecture with Integrating Reasoning	Hexmoor, 1993
Guardian		Hayes-Roth, 1990
HILARE (LAAS)		Alami, 1998
HOMER		Vere, 1991
JPL	Jet Propulsion Laboratory	Gat, 1990
KAoS		Bradshaw, 1997
LAAS (HILARE)		Alami, 1997
MAST (MIX)		González, 1995
MAX		Kuokka, 1991
NASREM	NASA Standard Reference Model	Albus, 1991
OAA	Open Agent Architecture	Cohen, 1994
Payton's*	Payton Architecture	Payton, 1986
PHOENIX		Howe, 1991
PRODIGY		Veloso, 1995
PRS	Procedural Reasoning System	Georgeff and Lansky, 1987
RALPH-MEA		Ogasawara, 1991
Reactive Deliberation		Sahota, 1993
REAKT		[Kersual, 1994][Mensch, 1993]
RETSINA		Lenox et al., 2000
Saphira		Konolige, 1996
SC-Agent	System Communication Agents	Posadas and Simo, 2002
SEC		Heck et al., 2001
SFX	Sensor Fusion Effects	Murphy, 2000
Sharp*		Laugier, 1999
SOAR		Laird, 1987
SSS	Servo, Subsumption, Symbolic	Connell, 1992
Subsumption	Subsumption Architecture	Brooks, 1986
Supervenience		Spector, 1992
TCA	Task Control Architecture	Simmons, 1994
Teleo-reactive		Benson and Nilsson, 1995
TETON		VanLehn, 1991
THEO		Mitchell,1991

8 ANEXO: Enlaces de software de soporte a las arquitecturas de control

Nombre	URL
ARIA	www.activmedia.com
ARS MAGNA	www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/testbeds/arsmagna
Autonomous Robotic Soccer	robotsimulators.8m.com/
Biorobotic	www.life.uiuc.edu/delcomyn/RSSimSoftware.html+C4
B-Soccer	members.fortunecity.com/wwann/#
BugWorks	www.cogs.susx.ac.uk/users/christ/bugworks/
CARMEN	www-2.cs.cmu.edu/~carmen/
DROS	dros.org/
DynaMechs	dynamechs.sourceforge.net/
Dynawiz XMR	www.concurrent-dynamics.com/xmr/
EASY-ROB	www.easy-rob.de/
EDTSim	www.enigmaindustries.com/
Evolving Robots	www.erachampion.com/ai/
EyeSim	robotics.ee.uwa.edu.au/eyebot/doc/sim/sim.html
eyeWyre	www.eyewyre.com/
FLAT 2-D Robot Simulator	www.cs.utexas.edu/users/qr/robotics/flat/
Gazebo	playerstage.sourceforge.net/gazebo/gazebo.html
Gwell	diablo.ict.pwr.wroc.pl/%7epjakwert/
Javabots	rogiteam.udg.es/docencia/iatm/javasoccer/edu/gatech/cc/is/docs/
Juice	www.natew.com/juice/
Khepera Simulator	diwww.epfl.ch/lami/team/michel/khep-sim/
Kinlab	kinlab.dyndns.org/
Map Viewer	mapviewer.skynet.ie/
MIARM	miarn.sourceforge.net/html/index.html
MissionLab	www.cc.gatech.edu/aimosaic/robot-lab/research/MissionLab/
Mobile Robot Simulator	jdlope.tripod.com/mrs.html
Mobility	www.irobot.com
MobotSim	www.mobotsoft.com/mobotsim.htm
MOBS	robotics.ee.uwa.edu.au/mobs/
MS Robotics Studio	msdn.microsoft.com/robotics
Nepumuk	asl.epfl.ch/~rolo/nepumuk.html
OOMRM	mysite.verizon.net/resowiky/simulation.htm
Open Automation Project	oap.sourceforge.net/
Open-R	open.aibo.com
OpenSim	opensimulator.sourceforge.net/
OROCOS Project	www.orocos.org/
POPBUGS	www.cogs.susx.ac.uk/users/christ/popbugs/intro.html
RARS	rars.sourceforge.net/
RoboCup Soccer Simulator	sserver.sourceforge.net/
RoboJDE	www.ridgesoft.com/robojde
ROBOOP	www.cours.polymtl.ca/roboop/
Robotect	www.ophirtech.com/products/
RobotFlow	robotflow.sourceforge.net/
Robotica	robotica.isa.upv.es/virtualrobot/
RobotWorks	www.robotworks-eu.com/
RoboWorks	www.newtonium.com/
RobSim	www10.brinkster.com/geniusportal/robsim.html

RobuBox	www.robosoft.fr
Ropsim (Camelots)	www.camelot.dk/
Rossum's Playhouse (RP1)	rossum.sourceforge.net/sim.html
RRG Kinematix	www.robotics.utexas.edu/rrg/downloads/software/rrgkmax4.0/
RWI	www.irobot.com
Saphira	robots.activmedia.com/Saphira/
Simbad	simbad.sourceforge.net/
Simderella	www.robotic.dlr.de/Smagt
SimRobot	www.informatik.uni-bremen.de/simrobot/win32_e.htm
Simulator BOB	simbob.sourceforge.net/
Stage/Player	playerstage.sourceforge.net/
Teambots	www.cs.cmu.edu/~trb/TeamBots/
ThreeDimSim	www.havingasoftware.nl/software/ThreeDimSim/ThreeDimSim.htm
TRSoccers	www.trsoccerbots.org/
VSOC	vsoc.sourceforge.net/
Webots	www.cyberbotics.com/
Yobotics	yobotics.com/simulation/simulation.htm