DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

DEPARTAMENTO DE COMPUTACIÓN
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
UNIVERSIDAD DE BUENOS AIRES

Tesis de Licenciatura en Ciencias de la Computación

Borges: una arquitectura para robots autónomos móviles

Javier Antonio Barra - jbarra@dc.uba.ar

LU: 501/95

Director: Dr. Juan Miguel Santos

Septiembre de 2010

A Rocío y a todos los que confiaron en mí.

Acerca de cómo se modifica el cerebro de las personas ciegas en la edición *National Geographic* en español de marzo de 2005 se lee en la página 27:

"Es provocador, pero estamos planteando que el cerebro podría no estar en absoluto organizado en modalidades sensoriales", dice Pascual-Leone. Lo que los neurocientíficos han estado denominando corteza visual en los últimos 100 años podría no estar dedicada exclusivamente a los ojos, sino que una manera más precisa de definirla sería como el área del cerebro que es capaz de discriminar relaciones espaciales, y va a tomar cualquier estímulo disponible para realizar esa tarea."

AGRADECIMIENTOS

En primer lugar quiero agradecer a mi director de tesis Dr. Juan Miguel Santos no sólo por su función de director sino también por su tolerancia, comprensión y confianza que depositó en mí. Sin él este trabajo nunca habría visto la luz. También quiero agradecer a mi hermana María Josefa por ayudarme a comprender algunas de mis propias ideas y por ser la compañera constante de este viaje, hecho que más estimo. Es imposible dejar de agradecer a mi madre y a mis hermanos María Josefa, Juan, Jorge, María Elena y María Jorgelina por ser cada uno ellos un ejemplo para mi además de las personas que más me han formado.

ABSTRACT

La propuesta de este trabajo es facilitar el diseño incremental de robots autónomos móviles. Para alcanzar este objetivo se ideó una arquitectura de robot y un framework de software basado en esta. El aspecto considerado para orientar el desarrollo incremental fue la estructuración del ambiente de actuación del robot. Así, el plan que implícitamente se propone aquí para el desarrollo de robots es comenzar con uno para un ambiente altamente estructurado para luego incorporar gradualmente más funcionalidad de modo que el robot finalice pudiendo operar en un ambiente no estructurado.

La arquitectura toma como cimiento para el diseño incremental un conjunto de principios conceptuales que se espera que atraviesen a todos los diseños del robot; desde aquellos iniciales, para ambientes estructurados, hasta los avanzados para ambientes generales. De este modo, la arquitectura plantea un conjunto principios conceptuales que declaran cuales deben ser los factores predominantes de un diseño para ser satisfactorio y un conjunto de guías que orienten el proceso de diseño hacia aquel fin.

Como resultado de los principios y guías arquitecturales se promueve utilizar un lenguaje de tipo declarativo para el desarrollo de las aplicaciones de los robots. La elección de este tipo de lenguaje es para permitir confeccionar las aplicaciones para los robots llevando a cabo una programación focalizada en el dominio de aplicación sin contemplar en este nivel de abstracción a los sensores y a los actuadores. Y, puesto que esta clase de lenguajes requieren de un framework subyacente, se delega en los algoritmos del framework la elección final de los sensores y de los actuadores con la intensión que eso brinde la flexibilidad que necesita un robot autónomo avanzado.

El modo de cómo atravesar distintos diseños de un modo uniforme es resuelto en la arquitectura al sustentarse en aspectos semánticos profundos de los robots autónomos. Los aspectos que conforman el denominador común semántico son: la estructura tridimensional del espacio, la caracterización de propiedades físicas -luminosidad, masa, etc-, el uso de entidades en el espacio y requerimientos explícitos de certeza y precisión para la información asociada a las entidades. Es por eso que se estima que los elementos creados bajo esta semántica son el aporte más significativo de este trabajo ya que, son el medio, de lograr una comunicación homogénea y orgánica tanto dentro de un mismo robot como entre los pertenecientes a una línea evolutiva.

En efecto, es posible identificar algunos de los problemas que se observan en otras arquitecturas existentes, en las que la ausencia de un denominador común semántico entre sus componentes ocasiona un excesivo aumento de complejidad cuando aumenta la funcionalidad del robot. Se advierten, principalmente, dos problemas. El primero, relacionado con el producto -el sistema robótico- en lo vinculado con la escalabilidad de su funcionalidad y el segundo, relacionado con el proceso -la construcción del producto- respecto a lo práctico que resulta llevar a cabo una construcción incremental. Estos dos problemas comparten la misma raíz en tanto cuentan con formas heterogéneas de interacción en su configuración interna. Debido a la ininteligibilidad recíproca entre las partes cuando se plantea aumentar la funcionalidad, con exigencias de uso de las partes distintas a las originalmente previstas, deben agregarse componentes adaptadores para lograr la comunicación interna del sistema robótico, aumentando, de ese modo, la complejidad de todo el sistema, y atentando así contra la factibilidad de un comportamiento automáticamente flexible.

En síntesis, se presenta en este trabajo una arquitectura y el diseño de un framework de software conformado bajo los fundamentos de la arquitectura, como así también, un caso de estudio para validar los beneficios de la arquitectura propuesta.

CAPÍTULO 1 - INTRODUCCIÓN

La construcción de un sistema robótico autónomo móvil aunque sea sólo la componente del software es una tarea sumamente compleja. Eso ha llevado a dividir el asunto en problemas o temas de interés más pequeños -coordinación, navegación, etc.- de modo que puedan ser tratados profundamente. Sobre estos tópicos en los últimos años se han alcanzado progresos importantes, sin embargo, aún no se llegado un acuerdo generalizado de cuál es el mejor modo de llevar a cabo la construcción de un sistema robótico autónomo.

En el principio dos puntos de vista extremos se establecieron alineándose en las arquitecturas deliberativas y reactivas; sin embargo, lejos de visualizarse en la actualidad un marcado sesgo hacia algunas de estas dos, todo parece indicar que las arquitecturas híbridas que combinan aspectos de ambas marcan el camino [7].

A pesar de que es poco probable que los sistemas robóticos autónomos avanzados se construyan exclusivamente bajo las directivas de alguna de las arquitecturas extremas estas tienen al menos como virtud que son claras de caracterizar [8]. En cambio, las arquitecturas híbridas son muy heterogéneas. Es por eso que se excluye una categorización de las mismas debido que, una mera enumeración de algunas de estas y descripción de sus particularidades, sería un proceso ocioso que no aportaría al entendimiento y se alejaría significativamente del propósito central de este trabajo que es conformar las guías para la construcción de un robot autónomo móvil.

No obstante, las arquitecturas híbridas observadas sirven para resaltar que el nicho conceptual propuesto en esta arquitectura no estaba siendo tratado. Básicamente, se notó la ausencia de un sentido orgánico conceptual que las definan; y más que arquitecturas constituyen una generalización de casos concretos en la cual se incluyen temas actuales de estudio. Los criterios de abstracción utilizados rememoran la forma en cómo los sistemas operativos o las redes de comunicación se construyeron desde el hardware hasta alcanzar a las aplicaciones [4][6]. Así, este enfoque que se observa en las demás arquitecturas las lleva a un resultado no orgánico ya que no consideran las diferencias significativas que los robots tienen con las otras máquinas con las cuales comparten el uso del hardware y del software.

Además, no sólo las arquitecturas son heterogéneas sino que también lo son las definiciones de las mismas. Por lo tanto, en este trabajo, se presenta una definición propia de arquitectura que, sin ser muy distinta a las que circulan en la bibliografía, es un reflejo claro de cómo se aborda el tema.

Un único requisito se estableció para la arquitectura. El mismo postula que la arquitectura debe facilitar a un grupo de personas desarrollar el software para un robot autónomo avanzado en forma incremental. Esto es, se debería poder comenzar con el diseño de un robot autónomo sencillo como objetivo y al mismo tiempo, este debiera ser el punto de inicio para un siguiente diseño más avanzado y así sucesivamente hasta alcanzar el robot buscado.

La mención del *grupo de personas* es importante ya que la cantidad de esfuerzo necesario para desarrollar las diversas técnicas que requiere un robot autónomo avanzado e integrarlas en un mismo sistema es mucho mayor de lo que una persona o grupo reducido pueden sostener antes de que la tecnología involucrada en el sistema se vuelva obsoleta.

Por el lado del diseño incremental se plantea la pregunta:

> ¿En qué aspecto se debe llevar a cabo un diseño incremental?

De ahí, que para poder responderla se abstrajo la constitución de un robot autónomo con tres dimensiones:

- la habilidad para desenvolverse en el ambiente.
- el comportamiento.

- la habilidad para modificar el comportamiento según las circunstancias.

La primera se refiere a la flexibilidad que tiene el robot para actuar en distintos ambientes. Esta dimensión trata precisamente sobre la estructuración y no estructuración del ambiente así como en la forma en que se utilizan los sensores y actuadores. La segunda dimensión de los robots autónomos es su comportamiento. Y por comportamiento se entiende a las tareas que lleva a cabo el robot y para las cuales fue diseñado. Si se observa esta dimensión separada de la primera se puede llegar a la conclusión que pueden existir robots con un comportamiento muy simple que podrían moverse en ambientes sumamente no estructurados -quizás un recolector de basura urbano- como así también a la inversa robots con un comportamiento sofisticado para ambientes estructurados -tal vez un robot de mantenimiento exterior de una estación espacial-. Por último, la tercera dimensión de los robots autónomos se refiere a la capacidad del robot de modificar autónomamente su comportamiento apartándose del que el diseñador definió originalmente.

Una vez caracterizados los robots de este modo se decide que la arquitectura haga foco en las dos primeras dimensiones y, en particular, que la dimensión para el diseño incremental sea la resaltada. Sobre la tercera dimensión la arquitectura no se explaya prácticamente nada pero tampoco cierra caminos.

La arquitectura se plantea con cinco principios arquitecturales que se refinan en dieciséis guías cuyo producto conceptual se podría destilar en:

- describir declarativamente la acción.
- representar el conocimiento del mismo modo que la acción.

Para validar la arquitectura se realiza el diseño detallado de un prototipo funcional del framework basado en las guías propuestas. Este framework se califica como prototipo, más allá de que ciertos componentes completos se consideren dados y que los algoritmos sean inocentes, por el hecho que el diseño se limita a robots de dirección diferencial con dos ruedas y con armazón de forma cilíndrica.
Estructuralmente el framework está constituido por seis sistemas que se vinculan entre sí por mensajes. Si bien los nombres de los sistemas pueden ser similares a los vistos en otras arquitecturas su conformación es muy distinta ya que prácticamente todos los sistemas principales intercambian un mismo concepto a procesar que se denomina *instantánea*. Es precisamente este diálogo en un mismo idioma lo que le brinda robustez a la arquitectura. Entre los elementos más elaborados del framework está la definición completa de un lenguaje declarativo 4gl para la especificaciones de las aplicaciones.
Como último paso en la validación de la arquitectura se presenta en el capítulo *Caso de estudio* una pequeña aplicación basada en un juego de fútbol. Finalmente, en las conclusiones, se incluyen las futuras líneas de trabajo derivadas de esta tesis para el desarrollo de un robot autónomo avanzado.

CAPÍTULO 2 - ARQUITECTURA

Destino de la arquitectura

El propósito de la arquitectura Borges es dar un marco sostén a un proceso eficiente de investigación y desarrollo para el software de alto nivel de robots autónomos móviles. Esto es imprescindible para la concreción de un robot autónomo avanzado por la necesidad de integrar distintos ensayos de técnicas sobre un mismo sistema robótico por un grupo de personas que puede ser fácilmente mayor a 30 miembros.
La propuesta de esta arquitectura se centra en trazar líneas conceptuales que atraviesen de forma gradual los diseños de robots autónomos que operan desde ambientes estructurados a no estructurados. La arquitectura logra esto al definir sectores de corte en los distintos componentes -a diferentes niveles de abstracción- de modo que sea posible reemplazarlos por otros más sofisticados sin necesidad de que todo el sistema robótico se vea comprometido. Así, este proceder modular es el que favorece el trabajo en equipo sobre un mismo sistema robótico ya que los distintos grupos de desarrollo que pudieran existir no se ven significativamente afectados entre sí ante los cambios.

La arquitectura Borges

La inmadurez propia de las arquitecturas de los robots autónomos a llevado a una proliferación de definiciones sin existir una que podría llamarse canónica y como muestra basta observar el panorama de definiciones que Arkin presenta en *Behavior-Based Robotics* [1].

> Una arquitectura son los sistemas de software y especificaciones que proveen lenguajes y herramientas para la construcción de *sistemas basados en conductas* -i.e. robots-.
>
> Una arquitectura es la disciplina que se consagra al diseño de robots individuales y altamente específicos a partir de una colección de bloques de construcción de software comunes.
>
> Una arquitectura es el diseño abstracto de unas clases de agentes: un conjunto de componentes estructurales en la cual la percepción, el razonamiento y la acción ocurren; la funcionalidad específica e interfaz de cada componente; y la interconexión topológica entre los componentes.
>
> Una arquitectura proporciona los principios para la organización de un sistema de control. Sin embargo, además de proporcionar la estructura, impone restricciones en el modo en que el problema de control puede ser resuelto.
>
> Una arquitectura describe un conjunto de componentes arquitecturales y cómo estos interactúan.

Es por eso que en este trabajo se establece la siguiente definición:

> *Una arquitectura es un conjunto de principios conceptuales que declaran cuales deben ser los factores predominantes de un diseño para ser satisfactorio y un conjunto de guías que orienten el proceso de diseño hacia aquel fin.*

En este contexto, la arquitectura Borges, propone un grupo de principios y de guías arquitecturales para la construcción de robots autónomos móviles.

Principios

A continuación se enumeran los cinco principios que fomentan una dirección a seguir para encarar la construcción de robots autónomos móviles. Luego de eso se explicará cada uno de ellos.

1. Alejarse de los absolutos y acercarse a los relativos.
2. Separar las necesidades de movimiento de las capacidades de sensado y actuación.
3. El entendimiento es movimiento.
4. Representar las estructuras y las operaciones conceptuales separadamente.
5. Unir dinámicamente.

Todos los principios actúan en lograr un robot flexible que pueda sobrellevar las situaciones potencialmente siempre cambiantes con las que podrían lidiar los robots autónomos móviles. Así el primero de estos principios está orientado a guiar la construcción de robot para que se sustente más en relaciones -en un sentido amplio- que en patrones absolutos predefinidos. Este principio se observa en la percepción, cuando se compara el elemento de interés con otros del fondo en vez de con una plantilla absoluta, en la sincronización asincrónica con la referencia a eventos tangibles más que en una señal de reloj, y en un mapa sin centro.
El segundo principio básicamente aisla cuan estructurado es el ambiente de actuación y cual es la calidad del hardware del robot. Esto permite una descripción más simple de la aplicación, menos propensa a errores pero principalmente posibilita que los sensores y actuadores necesarios puedan intercambiarse durante la ejecución de la aplicación brindando una libertad al sistema robótico que no tienen aquellos en que los sensores y actuadores están físicamente vinculados. Esta propiedad no sólo aplica a lo recién mencionado sino que también permite actualizar un sistema robótico concreto con nuevas capacidades de actuación y/o de sensado sin modificar nada de la aplicación; y, tras la actualización, la aplicación podrá usufructuar inmediatamente las nuevas facultades.
El tercer principio establece que así como se actúa sobre el mundo físico se debe actuar en el mundo abstracto. En sí este principio no está muy explayado en el presente trabajo debido a lo simple de la aplicación mostrada pero se nota que todo el *conocimiento* que dispone el sistema está representado exactamente de igual modo que a la manera de como se actúa. Más adelante, en el capítulo del framework, las entidades abstractas y virtuales y las evoluciones descriptivas y especulativas serán la materialización de este principio.
El cuarto principio postula que las estructuras -básicamente plantillas que describen las entidades del ambiente- y las operaciones -de un modo simplificado los comportamientos del robot- sobre aquellas deben representarse separadamente. Esto determina que las estructuras se construyan a partir de un conjunto de atributos definidos y que las operaciones deben escribirse en función de estructuras genéricas que satisfagan ciertas condiciones sobre sus atributos. De ese modo se permite que las operaciones sean más versátiles ya que las mismas pueden adaptarse a nuevas estructuras sin que sea necesario modificarlas. Además, esto también posibilita la combinación de operaciones para la creación de otras nuevas de un modo robusto puesto que es formalmente verificable si una operación puede trabajar o no con otra debido a que las condiciones sobre los atributos son comparables.
El quinto principio define el modo sugerido para unir todo lo que se ha separado. Son tan extensas las posibilidades a las que se puede encontrar un robot autónomo que

hace inválido poder contemplarlas a todas por adelantado. Así, aunque el camino sea más difícil, se debe demorar lo más posible la vinculación de los elementos separados hasta que se este en la situación concreta a resolver. De modo que sea la situación la que determine los criterios de los enlaces; en el framework, este principio se observa en el sensado, la actuación y en alguna medida en el lenguaje propuesto.

Guías

Las guías son un refinamiento de los principios con el fin de delinear un plan estratégico para encarar el diseño de un robot autónomo móvil avanzado. Las 16 guías presentadas no son ortogonales sino que están interrelacionadas para abarcar la visión propuesta por esta arquitectura.

1. **Modelo sintético**: Utilizar en la descripción de la aplicación del robot un modelo sintético del mundo.
2. **Entidad simple**: Conformar el modelo con un conjunto de entidades simples -básicamente sólo un identificador por cada una de estas-.
3. **Modelo liviano**: Sólo las entidades con las cuales esté operando el robot en un momento son incluidas en el modelo.
4. **Propósito de uso**: Representar explícitamente, y de un mismo modo en el modelo, la información espacio temporal de las entidades según su propósito de uso con el fin de que puedan ser tratadas por unidades de procesamiento específicas a cada tipo. Se definen tres propósitos:
    - Descriptivo: registra los hechos del mundo interpretados a través del sensado.
    - Especulativo: registra los supuestos sobre el mundo que la aplicación del robot incorpora con antelación para la circunstancia en que se encuentre el robot.
    - Prescriptivo: registra las necesidades sobre la modificación del mundo que la aplicación del robot genera.
5. **Sincronización asincrónica**: Sincronizar el funcionamiento de los componentes del robot asincrónicamente.
6. **Evento tangible**: Basar los eventos de la sincronización asincrónica en condiciones que verifiquen en qué posiciones espaciales se encuentran las entidades del propósito descriptivo.
7. **Confiabilidad explícita**: Representar explícitamente la confiabilidad sobre la posición e identificación correcta de las entidades.
8. **Requerimiento explícito**: Gestionar explícitamente en la aplicación del robot los requerimientos sobre la confiabilidad en la posición e identificación de las entidades.
9. **Reflexión constitutiva**: Representar al propio robot en el modelo con entidades.
10. **Normalización del mundo**: El mundo del robot se debe normalizar al espacio tridimensional con sus rotaciones -6 dimensiones en total-, el tiempo y un conjunto de propiedades físicas cada una con un dominio de valores definido.
11. **Aislamiento de sensores**: Describir la capacidad de sensado del robot como el hecho de medir una propiedad física espacialmente referenciada.
12. **Aislamiento de actuadores**: Describir la capacidad de actuación del robot como las fuerzas que las entidades del robot -actuadores y estructura- pueden aplicar y soportar sin romperse.
13. **Descripción normalizada**: Describir los prototipos para las entidades con los dominios de la normalización del mundo. Esto es, por sus propiedades físicas -incluye mecánica- y no como una relación directa con los sensores y los actuadores.

14. **Percepción declarativa**: Utilizar un componente reutilizable en distintas aplicaciones que informe la identificación y posición de las entidades en base a las capacidades de sensado, a la descripción física de los prototipos de las entidades y a los requerimientos sobre la calidad de información para las entidades descriptos por el lenguaje de la aplicación.
15. **Actuación declarativa**: Utilizar un componente reutilizable en distintas aplicaciones que defina la activación de los actuadores para realizar lo solicitado en la información prescriptiva en base a las capacidades de actuación, a la descripción mecánica de los prototipos de las entidades y a la información prescriptiva de las entidades.
16. **Centrado en lenguaje**: Utilizar un lenguaje organizado alrededor de las entidades, la gestión de los requerimientos y la confiabilidad de la información sobre las entidades para conformar la aplicación del robot.

A continuación se fundamentarán cada una de estas guías arquitecturales.

Fundamentación del enfoque propuesto

Modelo sintético

La arquitectura se centra en un modelo del mundo compuesto por entidades. Se califica como modelo sintético porque la aplicación del robot no puede tratar directamente con lo sensado sino que lo hace con entidades que son interpretadas a partir de un proceso de sensado. De este modo el contenido del modelo es límpido aunque podría no corresponderse con el mundo real. Sin embargo, esto último es posible de ser compensado por el hecho que esta formulación del modelo posibilita un tratamiento semántico del mismo. Esto significa que se puede integrar fácilmente información del dominio de aplicación al modelo ya que esta información, y la procedente desde el sensado, están expresadas ambas del mismo modo con entidades. El planteamiento semántico, en el caso de utilizar sensado activo, también brinda herramientas para separar el ruido de la señal en el sensado porque se puede cotejar con la referencia que implica la entidad supuesta. Además un modelo constituido con entidades permite que la aplicación del robot sea más breve ya que no debe tratar con el proceso del sensado y de la actuación. En cierto modo la presencia de un modelo ya calificaría a la arquitectura como deliberativa según los cánones vigentes pero, en este trabajo, se considera más el modo en que se usa el modelo y cómo se construye el mismo a que sí existe o no como medio de calificar a una arquitectura de un tipo u otro. De hecho, para plantear un posición extrema, en un típico robot reactivo donde dos fotoceldas controlan la dirección de un motor implica una interpretación del mundo -se acerca el depredador por ejemplo- y una síntesis -activar el motor hacia atrás- siendo esto también un modelo.

Entidad simple

Ante la necesidad de una definición de que significa que una entidad sea simple en este trabajo se establece que es simple si consume poca memoria. Es algo así como un puntero o una entrada a un índice. La principal motivo de mantener la representación de la entidades acotadas en el tamaño es posibilitar su procesamiento en tiempo real.

Modelo liviano

Por modelo liviano se entiende que en el mismo sólo están presentes las entidades con las cuales esté operando el robot en ese instante. En cierta medida el modelo liviano remeda el estado sensable en un robot reactivo y es su fin mantener el tiempo de respuesta del robot dentro del tiempo real.

Propósito de uso

El fin de la guía *propósito de uso* es que el sistema robótico pueda auto programarse. Es por eso que se declaran explícitamente los propósitos del uso de la información gestionada por los robots. Así esta declaración semántica sirve para que formalmente

y orgánicamente se pueda discriminar desde el código y por el código entre lo que se percibe del mundo, lo que es conocido del mundo con antelación y lo que es necesario de cambiar del mundo. Sin embargo, esta declaración no es meramente formal, sino que esta íntimamente relacionada con la acción por medio de los distintos procesos del framework que utilizan la declaración del propósito de uso para tomar decisiones de cómo llevar a cabo las acciones.

Sincronización asincrónica

Se puede establecer casi por definición, y claramente como supuesto de trabajo, que los robots autónomos estarán inmersos en ambientes cambiantes. Este carácter cambiante de los ambientes se manifiesta en las actividades que ocurren simultáneamente en estos y, muy probablemente, además a distintos ritmos. Más aun, es muy poco probable que las actividades que ocurran se repitan en el futuro exactamente con las mismas relaciones de ritmo. Así, este supuesto, hecha por tierra la elección para la arquitectura de una sincronización sincrónica, que en principio podría parecer más sencilla, ya que la predicción de los ritmos de las actividades del mundo a los cuales se debe acoplar el robot es imposible o sumamente difícil. En cambio, una sincronización asincrónica, posibilita un modo natural de acoplar las acciones del robot a las distintas actividades que en el mundo real pudieran ocurrir ya que bastaría verificar si se cumplen, o no, *eventos tangibles*.

Evento tangible

Se procurará que los eventos que controlan la sincronización asincrónica sean expresiones espaciales sobre las entidades descriptivas. De ese modo se logra una unión totalmente orgánica entre las actividades que ocurrieran en el ambiente de actuación del robot y el control de la sincronización de las acciones del robot.

Confiabilidad explícita

La guía de la confiabilidad explícita establece que la posición y la identificación de las entidades en cada propósito de uso tienen asociada una probabilidad que indica la certeza que se tiene sobre la exactitud de los valores. Esto es imprescindible para una programación declarativa ya que posibilita que los procesos del framework se comporten satisfactoriamente de acuerdo a la calidad de la información que dispongan. A su vez la programación declarativa se relaciona con las guías de percepción y actuación declarativa que siguen cuyo fin es brindar flexibilidad al robot.

Requerimiento explícito

Esta guía determina que tanto las solicitudes de sensado como las de cambio de posición de las entidades que realicen los procesos que llevan a cabo la aplicación del robot deben contener explícitamente la confiabilidad en la posición y en la identificación que necesitan estos procesos. La importancia de los requerimientos se observa en la asignación de los sensores y los actuadores de modo de alcanzar la posibilidad de obtener el mayor provecho de los disponibles. También es importante para que estos procesos puedan declarar sus justas necesidades. Un desajuste en las necesidades puede demorar un proceso que le bastaría un dato de baja calidad, así como generar estragos en procesos críticos que requieran datos de alta calidad y que son preferibles de no empezarlos a realizarlos si no se dispone la certeza necesaria.

Reflexión constitutiva

Esta guía es el sustento para que los robots se puedan integrar realmente con el mundo físico. Postula que el propio robot se debe constituir de entidades y que estas sean incorporadas al modelo. Esto posibilita la percepción y la actuación declarativa tal como se verán en las respectivas guías.

Normalización del mundo

Un robot autónomo se desenvuelve en el mundo físico y por lo tanto el espacio que internamente gestione se debe corresponder en lo mejor posible con el primero. Por más simple que sea el robot este se desenvolverá en el mundo físico y si se simplifica la representación interna también se debería hacer lo propio con las capacidades de

sensado y de actuación que estarán indefectiblemente en el mundo físico. Todas estas adaptaciones, por ejemplo para “aplanar” el mundo, conllevan a procesamientos específicos que pueden no ser uniformemente aplicados a través de todos los sistemas del robot. Por lo tanto, se utiliza en la arquitectura como común denominador un espacio de seis dimensiones de modo de permitir la flexibilidad de comunicación entre los procesos subyacentes del framework.
Además del espacio completo, la normalización también implica el uso de un conjunto de dominios de valores que representan las propiedades físicas. Estos dominios constituyen una nítida interfaz entre el mundo físico y el modelo del robot y son centrales para las próximas guías.

Aislamiento de sensores

El fundamento para aislar los sensores de la aplicación del robot es para permitir intercambiar automáticamente de sensores de acuerdo a las necesidades de sensado del momento. El aislamiento se sustenta en que las entidades sean descriptas por propiedades físicas -i.e. la guía de descripción normalizada- y en que la capacidad de sensado del robot sea especificada por la facultad de medir propiedades físicas en el espacio. Por lo tanto, esto posibilita una vinculación indirecta entre las entidades y los sensores a través de las propiedades físicas que es realizada por el framework. Además la forma en cómo se realiza el aislamiento de los sensores también puede ser utilizado por el framework para llevar a cabo un sensado activo y/o la fusión de sensores.

Aislamiento de actuadores

La descripción de la capacidad de actuación establece que, como mínimo, se deben describir las fuerzas que el robot puede transmitir al medio, los elementos estructurales que transmiten las fuerzas dentro del robot y las limitaciones mecánicas que romperían los componentes involucrados. En cierto modo, esta descripción, es análoga a la del sensado ya que las fuerzas en su contexto espacial son primordiales. Si un robot ha de ser autónomo debe poder estimar la fuerza que necesitaría para cometer su fin -por ejemplo mover o detener una entidad- y una vez hecho eso debe lograr realizar una composición de fuerzas que se aproxime a la fuerza necesaria. El sentido espacial de las fuerzas es central ya que necesitará obtener los puntos de aplicación y de apoyo.
Así, con esta interpretación, el robot no tiene actuadores sino que tienen medios para generar fuerzas en puntos del espacio. El sentido de este enfoque es que como en el diseño no se pueden predecir todas las situaciones con las que se podría encontrar el robot entonces una solución general que funcione de acuerdo a esta interpretación debe estar disponible.

Descripción normalizada

La descripción normalizada se refiere a que todas las entidades sean descriptas por medio de propiedades físicas de igual modo a como se representen la capacidad de sensado y de actuación del robot. La descripción normalizada es imprescindible para un sensado activo flexible ya que permite seleccionar los sensores e intercambiarlos en base al cotejo del dominio de valores de la propiedad física en cuestión. Por el lado de la actuación, la descripción normalizada permite que se pueda seleccionar actuar en el mundo de acuerdo a las propiedades físicas; por ejemplo, a la fricción de la superficies o la masa de los objetos, operación sumamente conveniente para la obtención de los puntos de apoyo de la actuación así como para la capacidad de utilizar herramientas.

Percepción declarativa

La percepción declarativa está fuertemente motivada por el objetivo de lograr un robot de comportamiento flexible que pueda adaptarse a fallos en el hardware y a la utilización eficiente de los recursos de sensado que disponga. Esto es porque se espera que en un robot autónomo la gestión del sensado no sea sencilla ya que para deshacerse de situaciones ambiguas suelen ser requeridos múltiples sensores. Por eso esta guía plantea que el modo de alcanzarlo es que la aplicación que controla el robot no se comprometa tempranamente con determinados sensores sino que sólo declare las

necesidad de información sobre una entidad. En dicha declaración se detalla los requerimientos que debe satisfacer la información sobre la certeza de la identificación de la entidad así como sobre la certeza sobre la posición de la misma. Por eso es que se sugiere un componente reutilizable, separado de las distintas aplicaciones que pudieran escribirse para el robot, que se encargue de gestionar la percepción e informe la identificación y posición de las entidades. Este componente operaría en base a las capacidades de sensado, a la descripción física de las entidades y a los requerimientos sobre la calidad de información descriptos por la aplicación.

Actuación declarativa

Una vez más la actuación sigue una línea de razonamiento análogo a la de la percepción. Siendo por tanto el objetivo obtener flexibilidad en el actuar y, un modo de lograrlo, es que la aplicación no se comprometa tempranamente en cómo generará la fuerza necesaria para alcanzar su meta de acción.

La aplicación del robot hará uso del propósito prescriptivo para describir declarativamente las acciones que necesite que se produzcan en el ambiente. Las declaraciones de acciones incluyen requerimientos para considerar en la ejecución con los actuadores. Además, puesto que la aplicación se desacopla de los actuadores es que se recomienda para la actuación utilizar un componente reutilizable entre las distintas aplicaciones que defina la selección y la activación de los actuadores. Este componente tendría como función convertir lo solicitado en la información prescriptiva en la acción del robot y se basaría en las capacidades de actuación, en la descripción mecánica de la entidades y en la información prescriptiva de las entidades.

Centrado en lenguaje

Muchas de las guías presentadas se dirigen a poder sustentar la constitución de un lenguaje exclusivo para describir las aplicaciones de los robots. Tal idea se vislumbra cuando se observa que la aplicación no tiene que tratar con actuadores y sensores y se limita a operar sobre entidades. En particular la aplicación sólo declara necesidades de información sobre entidades, genera nuevas posiciones meta para las mismas y coordina su funcionamiento por medio de condiciones sobre la posición de la entidades. Es así que se advierte que todo el universo de aplicaciones puede reducirse a escasos elementos vinculados con pocas reglas que es lo que posibilita pensar en un lenguaje. Sin embargo, el real potencial del lenguaje nuevo con respecto a los genéricos es que se puede plasmar en este, y de un modo robusto, las restricciones propias que se suscitarían en esta clases de aplicaciones. Como un objetivo lejano, el ideal de lenguaje buscado debería ser uno reflexivo de modo que sea posible que el robot modifique su propio código a medida que gana experiencia.

CAPÍTILO 3 - FRAMEWORK

Framework Borges Uno

En esta sección se describe el diseño del framework Borges Uno que se sustenta en las guías arquitectónicas de la arquitectura Borges. El propósito del mismo es validar la arquitectura Borges.

El framework planteado es utilizado básicamente a través de un lenguaje declarativo y de tipo 4gl cuyo propósito es hacer más sencillo el desarrollo del software de aplicación para robots autónomos móviles. En este trabajo se entiende por *declarativo* al hecho de que el lenguaje describa *qué* debe hacer el robot pero no, *cómo* deba llevarlo a cabo ya que, esto último, estará oculto en la implementación del lenguaje-framework que lo realizará automáticamente con sus algoritmos. Se comprende por lenguaje de tipo 4gl a aquellos tipos de lenguajes que están orientados a un dominio de acción en particular y que incorporan, dentro de su sintaxis, aspectos semánticos al definir bloques de agrupación de código, estructuras de control y de datos ajustados específicamente a la semántica del dominio de aplicación.

El lenguaje del framework opera sobre un modelo del mundo y este modelo es gestionado por dos de los sistemas del framework: el sistema de integración espacio temporal (SIET) y el sistema de modelización de objetos (SMO). El primero se encarga principalmente de las relaciones entre las entidades, en particular, los vínculos espacio-temporales y de propósitos de uso. Aunque también gestiona la asignación de los recursos de hardware y los datos específicos asociados a cada entidad. El SMO tiene como fin categorizar el universo de entidades que podrían existir en el modelo. Eso se lleva a cabo mediante la utilización de *prototipos* los cuales registran la definición de las entidades. Los prototipos se clasifican -en función de si las entidades derivadas de los mismos pueden o no ser sensadas y de si ocupan o no una posición en el espacio- en tres tipos: abstractos, si sus entidades no pueden ni ser sensadas ni ocupar un lugar en el espacio; virtuales, si no pueden ser sensadas pero ocupan en cambio una posición en el espacio y concretos, si pueden ser sensadas y además ocupan una posición en el espacio. Por ejemplo, un prototipo abstracto podría ser un *score* que utilice la aplicación de algún modo y pueden asimilarse a una tupla de datos; los virtuales se utilizan mayormente para representan puntos, zonas o volúmenes del espacio que la aplicación utiliza para referenciar el espacio en su algoritmos y, finalmente, los prototipos concretos que son los objetos reales del mundo como podría ser una pared. En particular, los prototipos concretos tienen una descripción de cómo están constituidos físicamente para que en el nivel de implementación semántica del lenguaje pueda, por un lado, sensarlos y, por otro lado, actuar sobre ellos en forma automática a través de algoritmos.

Como ya se mencionó el propósito del lenguaje de la arquitectura es describir una aplicación para el robot. La misma puede ser observada desde dos puntos de vista dependiendo si se hace hincapié en la declaración o en la ejecución por parte de la implementación del lenguaje. Con el foco en la declaración, una aplicación es un conjunto de prototipos de las entidades que podrían estar en el modelo del mundo y un conjunto de operadores que describen cómo modificar los atributos de las entidades basadas en los prototipos. En cambio, desde una perspectiva situada en la ejecución, los prototipos se instancian en entidades del modelo y los operadores, en un conjunto de procesos concurrentes en comunicación entre sí que comparten las entidades del modelo -aunque cada uno contiene también su estado privado-. En este trabajo se definió cambiar la palabra operador por la de *evolución* ya que estos operadores describen la evolución temporal de entidades y llamar *recorrido* a los procesos que surjan de la evoluciones ya que determinan un camino posible de todos los potenciales que incluyen la evolución.

El código del lenguaje está organizado de modo tal de que las evoluciones y los prototipos estén separados en secciones distintas del código. Esta división se debe a que la arquitectura propone utilizar como ladrillo de construcción de las aplicaciones a las evoluciones las cuales podrían tener indefinido sobre que prototipos estas actuarán hasta que se las instancia en un recorrido con entidades. Por eso la definición de los prototipos puede ser leída por los recorridos y esta capacidad es utilizada para verificar si una entidad de un determinado prototipo satisface los requerimientos de la evolución del recorrido en cuestión. Por ejemplo, una evolución *empujar <entidad concreta>* no tiene sentido completo hasta que se instancia en un recorrido asignándose un entidad. Así, *empujar pelota*, en cambio, sí tiene un sentido completo.
Otro enfoque fundamental propuesto en la arquitectura es considerar al robot en sí mismo como una entidad o grupo de entidades manteniéndose la uniformidad de que cada una es derivada de un prototipo lo cual permite que no haya diferencias significativas entre mover el robot o un objeto del mundo. Una consecuencia interesante de esto es que establece un terreno firme sobre el cual construir –en una aplicación o como una ampliación posterior de arquitectura- una forma flexible tanto para que el robot pueda utilizar herramientas que pudiera encontrar en el medio así como para abstraer comportamiento desde otros agentes a través de la imitación.

El resto de este capítulo está compuesto por una sección por cada uno de los sistemas que conforman framework propuesto (Fig. 1). Así seis sistemas serán descriptos y cuyos objetivos son:

- Sistema de integración espacio temporal (SIET): El objetivo del sistema de integración espacio temporal es devolver una visión integrada y sintética del mundo a los sistemas que le proveen de datos para construirla.
- Sistema de modelización de objetos (SMO): Centralizar la modelización de los prototipos de las entidades y los servicios de información sobre los mismos con el fin que los demás sistemas puedan procesar las entidades sin la necesidad de disponer de cómo están constituidas.
- Sistema de gestión del hardware (SGH): Gestionar el uso de las entidades propias del robot del tipo actuador y sensor para que los demás sistemas no tengan que considerar las particularidades del hardware.
- Sistema de motorización del lenguaje (SML): Plasmar el motor del lenguaje declarativo 4GL con el cual se interpreta el programa de la aplicación.
- Sistema de percepción (SP): El objetivo del sistema de percepción es reconocer las entidades y determinar la posición en el espacio de las mismas..
- Sistema de actuación (SA): Convertir la declaración del movimiento para las entidades en comandos de acción concreta sobre los actuadores.

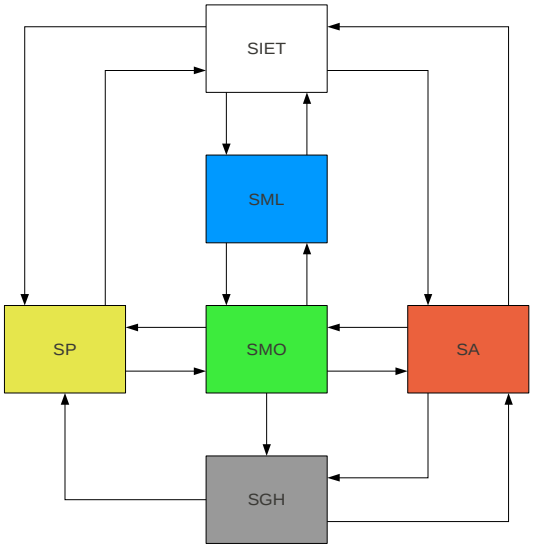


Figura 1 - Relación entre los sistemas constituyentes del framework Borges Uno.

Sistema de integración espacio temporal (SIET)

Objetivo

El objetivo del sistema de integración espacio temporal es devolver una visión integrada y sintética del mundo a los sistemas que le proveen de datos para construirla.

Introducción al SIET

En la presente sección se describe principalmente las funciones utilizadas por el SIET para cumplimentar con su objetivo y se describen algunos aspectos de una posible implementación de la funcionalidad presentada [2][3][4][6][11]. Se inicia el mismo con una explicación de los conceptos básicos -algunos ya introducidos brevemente en el primer capítulo- con el propósito de establecer un sólido vocabulario común.
Finalmente, se recuerda que la presente arquitectura establece contar con un modelo del mundo el cual se constituye principalmente en el SIET -el resto del modelo se establece en el sistema de modelización de objetos-. El modelo del SIET se divide en dos partes: un mapa y un escenario. El mapa mantiene las relaciones espaciales de las entidades involucradas en la historia persistente del robot en forma atemporal, es decir, el mapa se mantiene después del reinicio del robot. En cambio el escenario registra las relaciones espacio temporales involucradas en las acciones del robot que haya acabado de ejecutar, que esté ejecutando o que esté próximo a ejecutar. Por estar íntimamente relacionado con las acciones el escenario es volátil, lo que significa que, tras el reinicio del robot, aquel no se mantiene. Finalmente en apéndice "Interfaces entre sistemas" se detalla la conexión del SIET con los demás sistemas de la arquitectura.

Definición de conceptos

Entidad

La entidad es el elemento sobre el cual se organiza el modelo del SIET. Básicamente para el SIET una entidad es sólo un identificador que representa una instancia de un prototipo -ver la sección "Sistema de modelización de objetos"- y su función es ser el punto de acceso a los elementos que sí guardan información. Si bien existen varias clases de entidades -concretas, virtuales y abstractas- la arquitectura establece que habrá una entidad concreta llamada *Self* que denota al propio robot cuyo propósito es básicamente establecer un marco de referencia espacial para las demás entidades.
Para el modelo el robot está construido no sólo con *Self* sino además por otras entidades que se clasifican principalmente en *sensor*, *actuador* y *estructural*. A las entidades sensor y actuador se las llama globalmente también como *recursos*. Todas las entidades que pertenecen al robot son clasificadas como *propias* y, por complemento, todas aquellas que no conforman el robot -y que por lo tanto son los objetos del ambiente- son llamadas *no propias*.

Espacio del SIET

La definición de espacio que utiliza el SIET para su modelo es aquella que consiste de las siete dimensiones clásicas necesarias para posicionar un objeto en el espacio y en el tiempo siendo estas: *rumbo*, *altura*, *distancia*, *cabeceo*, *alabeo*, *guiñada* y *tiempo*. No obstante, además del concepto de *posición* que consiste en utilizar las siete dimensiones recién citadas también se usan los conceptos de *ubicación* para señalar sólo las dimensiones *rumbo*, *altura* y *distancia* y el de *actitud* para indicar solamente las de *cabeceo*, *alabeo* y *guiñada*.

Tolerancia y confiabilidad

La tolerancia es un atributo de las dimensiones que indica en cuánto puede variar el valor de la dimensión. La confiabilidad es a su vez un atributo de la tolerancia que indica la probabilidad de que el valor real de una dimensión esté dentro del margen establecido en la tolerancia.

Instantánea

Las instantáneas son el modo en que se representan las distintas posiciones *"instantáneas"* que una entidad puede tomar para una *fuente*; concepto, este último, que se definirá en breve. Las instantáneas son generadas siempre por una única fuente y distintas fuentes pueden compartir una misma entidad y por lo tanto establecer, para un mismo instante, que la entidad compartida ocupe posiciones distintas. Claramente esta situación no es consistente y es función del SIET resolverla.

Toda instantánea tiene un identificador único que es utilizado con dos fines: el primero por la fuente que la haya generado, para poder hacer referencia a esta a posteriori y así actualizar los atributos de la instantánea que generó y, el segundo por el mecanismo de sincronización, para poder etiquetar las instantáneas como se verá más adelante en la definición del concepto "Restricción de ejecución".

Fuente

Como ya se mencionó el SIET tiene comunicación con otros sistemas pero, si se es más específico, en realidad está en contacto con los recorridos del sistema de motorización del lenguaje y con las unidades procesadoras de los sistemas de percepción y de actuación -que en sus respectivas secciones se detallarán-. Puesto lo heterogéneo de este escenario, para el SIET, todos estos elementos recién citados son llamados colectivamente como *fuentes*. Básicamente una fuente es una unidad procesadora cuyas funciones, desde la perspectiva del SIET, son: determinar qué entidades necesita que le provea el SIET, establecer los requerimientos para la obtención de datos de sus entidades, recibir las instantáneas descriptivas y especulativas desde el SIET, generar instantáneas para ser entregadas al SIET y finalmente cancelar el uso de las entidades que había necesitado.

Propósito de uso

Las fuentes pueden usar a una entidad con distintos propósito de uso. Por ejemplo, una fuente puede necesitar saber dónde está una entidad y otra fuente -o la misma inclusive- puede necesitar mover la entidad presentándose así, en este ejemplo, dos usos de una entidad.

En la sección "Sistema de motorización del lenguaje (SML)" se profundiza el significado de los distintos propósitos de uso, sin embargo, aquí se adelanta que existen tres tipos posibles: especulativo, descriptivo y prescriptivo. Estas tres categorías se ven reflejadas en el SIET a través de tres tipos de instantáneas distintas para las entidades. Para el SIET, los tipos de instantáneas determinan distintas restricciones formales en la construcción del modelo con el fin de representar la semántica que las fuentes les otorgan a cada uno de estos tres propósitos de uso.

Instantáneas prescriptivas declarativas y procedimentales

Las instantáneas prescriptivas declarativas son las que generan el SML y el SP. Estas declaran los movimientos que se precisan de las entidades. En cambio las instantáneas prescriptivas procedimentales son generadas sólo por el SA y solamente aplican a entidades propias de tipo actuador. Esto es, sólo describen los movimientos válidos de los actuadores del robot.

Requerimientos para entidades

Además de las instantáneas existe un elemento más que depende de la entidad, de la fuente y del tipo de propósito de uso siendo este elemento el *requerimiento*. Así, un

requerimiento de una fuente para un propósito de uso de una entidad determina las restricciones de los valores de las dimensiones que deben cumplir las instantáneas de aquella entidad. Se incluye también en el requerimiento el grado de identificación del prototipo de la entidad y de la identidad de la entidad.
Los requerimientos son elementos relativamente estables ya que cuando son establecidos por su fuente los mismos persisten para todas las subsiguientes instantáneas de la entidad hasta que la fuente constituya un nuevo requerimiento que reemplace al anterior. Así, puede haber a lo sumo y en todo momento, tres requerimientos por fuente y entidad, uno por cada tipo de propósito de uso.

Materialización

Por materialización se llama al proceso que se lleva a cabo entre el SIET, el sistema de percepción (SP) y el sistema de actuación (SA) para lograr sensar las entidades a partir de sus requerimientos o moverlas de acuerdo a sus instantáneas prescriptivas. Así con la materialización se crean los movimientos reales que aplicaría el robot.

Para el caso del SP una simplificación de los pasos del proceso es:

- Recibir los requerimientos para las entidades desde el SIET.
- Seleccionar los sensores a utilizar.
- Generar instantáneas prescriptivas sobre las entidades de los sensores si el SP decidió realizar un sensado activo.
- Enviar al SIET las instantáneas prescriptivas de los sensores para que este se encargue de que se materialicen.
- Generar instantáneas descriptivas para las entidades que pudo determinar su posición e identificarla a través del sensado.
- Entregar aquellas instantáneas descriptivas al SIET.

En cambio para el SA una simplificación consiste en:

- Recibir desde el SIET las instantáneas prescriptivas declarativas de las entidades tanto no propias como propias.
- Seleccionar los actuadores que usará.
- Generar un conjunto de instantáneas prescriptivas procedimentales para las entidades propias de tipo actuador con la intensión de que el robot aplique fuerzas sobre las entidades objetivo y así estas últimas describan lo establecido por las instantáneas prescriptivas declarativas.
- Entregar al SIET las instantáneas prescriptivas propias procedimentales para que las integre.

Ejecución

La ejecución se llama al proceso que lleva a cabo el SIET, el SA y el SGH que consiste:

- verificar si la asignación de los recursos de hardware solicitados en la materialización presenta conflictos,
- resolver los conflictos presentados,
- medir la performance de las fuentes.

Una simplificación de los pasos del proceso son:

- Que el SIET verifique si las instantáneas prescriptivas procedimentales de entidades propias presentan conflictos.
- Que el SIET seleccione entre las instantáneas anteriores cuales resuelven el conflicto y envíe al SA el conjunto de los identificadores que representan a estas instantáneas.

- El SA recibe los identificadores, identifica las instantáneas, que previamente había generado y enviado al SIET a integrar, y genera y envía al SGH los comandos de actuación.
- El SGH recibe los comandos, los ejecuta en el hardware y envía una respuesta al SA.
- El SA con las respuesta analiza el desempeño de la ejecución de cada fuente y le envía la conclusión al SIET.

Restricción de ejecución

Las instantáneas generadas por las distintas fuentes tienen que ser ejecutadas en un orden determinado para que el comportamiento del robot sea coherente. Si bien en el capítulo de la arquitectura se estableció la guía de los eventos tangibles la utilización de un evento para sincronizar cada instantánea es un despropósito. Por eso, las instantáneas se sincronizan por omisión por el evento que es la ejecución de instantáneas precedentes. Así cada instantánea lleva un conjunto de identificadores de instantáneas llamado restricción de ejecución. El propósito de la restricción de ejecución es que una instantánea no será ejecutada hasta que todas las instantáneas cuyos identificadores estén en su restricción de ejecución hayan sido ejecutadas. Los eventos tangibles funcionan por arriba de este mecanismo y especifican condiciones que, hasta que no se cumplan, evitan que ciertas instantánea sean ejecutadas. De modo que con la verificación de la ejecución de instantáneas y con provocar la demora de la ejecución hasta que se satisfaga la condiciones de los eventos tangibles se establece la sincronización del robot. Finalmente, si la restricción de ejecución está vacía significa que la instantánea en cuestión no tiene restricciones de precedencia con respecto a otras.

Instantáneas pasadas, presentes y futuras

La clasificación en instantáneas pasadas, presentes y futuras es meramente una convención práctica útil para describir el estado del escenario ya que el SIET no distingue a las instantáneas por esta clasificación. La definición se construye en base a las restricciones de ejecución que vinculan las instantáneas que genera una fuente. Así, puesto que toda instantánea -sea del propósito de uso que sea- se vincula con al menos una instantánea -la anteriormente generada por la fuente[1]- entonces se define que, dada una instantánea, las del pasado serán sus precedentes y las del futuro serán las instantáneas que la suceden. Sólo falta mencionar que la instantánea presente de una fuente es la que se esté ejecutando en cada momento.

La dimensión tiempo

Un detalle importante es que el tiempo es la única dimensión que tiene dos usos. El primero relacionado con el dominio de aplicación del robot para organizar la interrelación de las instantáneas a lo largo del tiempo y el segundo relacionado con el costo en tiempo de las computaciones. Son llamados respectivamente *time* y *processtime*. Un punto a detallar sobre ambos es que en sus requerimientos se incluye un atributo más para indicar una frecuencia de actualización para que así las fuentes no tengan que actualizar el requerimiento sólo por el transcurso del tiempo.

Time

Una primera aclaración sobre el atributo *time* es que el mismo no se refiere ni es utilizado para la sincronización de eventos, pues como ya se mencionó, la sincronización es asincrónica. En cambio el atributo *time* es usado para establecer un marco de referencia tentativo para poder comparar a qué momentos pertenecen las instantáneas de distintas fuentes.

[1] Esto no es válido para la primera instantánea generada por una fuente en cuyo caso la restricción de ejecución es vacía.

Como se describió previamente el sentido temporal de pasado, presente y futuro está referido a las restricciones de ejecución con respecto a la instantánea presente; sin embargo, puesto que las fuentes llevan cada una su propio ritmo y las instantáneas que generan pueden no estar vinculadas con una restricción de ejecución se presentan situaciones en las que no se podría discernir que suceso ocurre antes o después. Esta necesidad se presenta cuando es necesario construir por parte del SIET una visión integrada del futuro y es por eso que no es necesario que el valor del atributo *time* sea muy exacto dado que la sincronización no descansa sobre este sino que lo hace sobre las restricciones de ejecución. Toda fuente genera sus instantáneas para un rango del atributo *time* sin dejar un espacio del dominio del mismo sin cubrir. Por eso se califican como abiertos los rangos de la primera instantánea y de la última siendo además en las únicas instantáneas en que están permitidos dichos rangos abiertos.

Processtime

El *processtime* es formalmente un requerimiento más pero, dada la importancia intrínseca del tiempo de procesamiento real es que se le asigna un nombre propio. El *processtime* establece un período dentro del cual tiene sentido para la fuente que se ejecute la instantánea prescriptiva que generó o el requerimiento que solicitó; más allá de ese período, la instantánea y/o el requerimiento serán descartados por el SIET y se avisará a la fuente sobre ese hecho.

Importancia de fuente

La importancia de una fuente es la posibilidad que tiene una fuente para que sus instantáneas puedan adquirir más prioridad que otras para ser ejecutadas. Esto significa que las instantáneas de la fuente pueden tener mayor prioridad que otras cuando la fuente lo necesite en forma inmediata pero que en muchos otros cederá esa prioridad a instantáneas de fuentes de menor importancia. La importancia se define en base al orden que surge de recorrer[2] el árbol de activación de recorridos del sistema de motorización del lenguaje (SML) en la forma “a lo ancho primero” -BFS-. Para aquellas fuentes que no son recorridos -las unidades procesadoras del SP y del SA- la importancia la obtienen a través de los *pedidos* que llegan al SP y al SA desde SML mediados por el SIET. En sí su importancia será un número fraccionario entre el orden del recorrido anterior y siguiente. Así, las instantáneas generadas por el recorrido *Main* -ver sección “Sistema de motorización del lenguaje (SML)” pag. 77- son las más importantes de todas.

Índice de ejecución

El índice de ejecución de la fuente -ver sistema de actuación (SA)- señala cómo ha sido favorecida en el *processtime* una fuente considerando la relación entre el momento de ejecución esperado por la fuente y el momento en que efectivamente se ejecuta -si no vence-. El índice varía entre -1 y 1. Un valor cercano a 1 indica una fuente en que sus instantáneas son casi siempre inmediatamente ejecutadas, en cambio, un valor próximo a -1 señala una fuente en la cual sus instantáneas tienen que esperar mucho para ser ejecutadas pudiendo inclusive llegar al caso de que deban ser descartadas sin materializar.

Hitos de interrupción

Básicamente está relacionado con el carácter reactivo o planificado de una fuente. Un hito determina un punto en una serie de instantáneas de una fuente -un plan- en que el plan puede ser interrumpido. En las fuentes reactivas cada instantánea pertenece a un plan continuo y no tienen por lo tanto hitos de interrupción.

[2] Aquí se presenta un caso confuso de terminología. Cuando se utiliza el verbo *recorrer* se refiere al algoritmo que se aplica sobre una estructura de datos de árbol en cambio la uso de la palabra *recorrido* como sustantivo se refiere a las instancias que se crean de las evoluciones del SML.

Configuración de hardware

La configuración de hardware es una abstracción para referirse a los distintos modos de funcionamiento que pueden tener los sensores y actuadores del robot. Las configuraciones de hardware son definidas en el sistema de modelización de objetos -SMO- en base a los drivers que brinda el sistema de gestión de hardware -SGH-. Por medio de los drivers el SGH oculta el hardware del robot al resto de la arquitectura.

Organización interna

Tal como ya se adelantó el modelo del SIET consta de dos partes: un mapa y un escenario. El mapa registra las relaciones espaciales en forma atemporal de las entidades involucradas en la historia persistente del robot, es decir, el mapa se mantiene después del reinicio del robot; y el escenario, en cambio, es volátil ante el reinicio y refleja la parte del mapa donde el robot esté ejecutando sus acciones. Mientras el mapa es sólo topológico el escenario es también temporal cubriendo un breve período de tiempo hacia el pasado y para el futuro. Asimismo el escenario es el soporte al mecanismo de sincronización y la asignación de los recursos de hardware. El mapa y el escenario están unidos sin solución de continuidad pero no temporalmente, sólo el escenario maneja el tiempo y por ende es también en el escenario donde se lleva el registro del micro plan del robot.

La lista de los principales ítems registrados en el modelo es:

- Identificadores de fuentes que se corresponde con los procesadores internos del SA y del SP y con los recorridos del SML.
- Requerimientos para las instantáneas desde las fuentes.
- Entidades del SML.
- Instantáneas prescriptivas presentes -reacción- y futuras -planificación- generadas por el SP y SML.
- Instantáneas descriptivas presentes sensadas por el SP -estímulo para reaccionar-.
- Instantáneas descriptivas presentes -estimación- y futuras -predicción- generadas por el SML.
- Las instantáneas especulativas del SML.
- Las instantáneas prescriptivas procedimentales del SA.
- Las relaciones espaciales y temporales -sincronización- entre las instantáneas.
- Recursos y los pedidos y las asignaciones de los mismos a las instantáneas.

A continuación se describen algunas ideas para una posible implementación del modelo del SIET. Básicamente lo propuesto es un grafo sobre varias clases de nodos y de arcos junto con un conjunto de restricciones de construcción. Se detallan los conceptos involucrados y la forma de construir y de mantener el modelo antedicho.

Representación espacial

La acción de fijar que representación espacial es conveniente para el modelo es un análisis que se realiza con respecto al costo computacional de tratarla de acuerdo a la manera en que se utilice la información registrada en el mismo. En particular se analiza el marco de referencia cubriendo dos aspectos: la ubicación del punto de referencia, que puede ser ego referenciado o topo referenciado, y la cantidad de puntos de referencia utilizados, siendo un punto para un esquema absoluto y varios para uno relativo. Puesto que tanto el escenario como el mapa comparten la representación espacial no se realizará distinción entre ambos.

Elección de sistema

La conveniencia o no de una representación espacial está en función de qué datos sean registrados en el modelo y qué tareas deban llevarse a cabo con soporte en aquel. Así, en un modelo con pocos hitos espaciales registrados, estos aspectos se desdibujan para hacerse sólo relevantes cuando el tamaño del modelo sea considerable. Puesto que la arquitectura está orientada a un robot autónomo se presupone que su espacio de actuación puede ser extenso y, a costa de tener una representación más compleja de lo necesario para los modelos pequeños, es que se decide utilizar una representación espacial que se adecue a grandes modelos.

La performance de implementación de varias funciones son sensibles a la representación del espacio que se utilice pues de este depende si habrá que llevar a cabo conversiones espaciales o no. Por ejemplo, el sensado y la actuación son inherentemente egocéntricos pero la planificación del desplazamiento y orientación en el espacio son topológicos.

Por el lado de la cardinalidad de puntos de referencia, cuando el modelo es pequeño un único punto absoluto puede ser conveniente pero a medida que el tamaño del modelo sea mayor más importantes serán los problemas de escala ya que se requerirá una mantisa grande para poder representar algo con precisión de milímetros cuando está a varios kilómetros del origen de coordenadas.

Como se mencionó un punto de referencia egocéntrico es adecuado para el sensado y la actuación y se ajusta mejor para un robot reactivo que no utiliza un mapa. Sin embargo, si se pretende utilizar un mapa con aquel esquema cada vez que se mueva el robot se habrá de ajustar las coordenadas de todos los hitos del mapa. Cuando el mapa es pequeño esto no es muy costoso pero sí si el mapa es grande.

Asimismo un esquema topológico con un único origen absoluto es conveniente para un robot deliberativo que realice planificaciones ya que cada vez que se desplace no se tendrá que actualizar todo el mapa sino sólo la posición del robot en el mismo. Inversamente al caso anterior, cada vez que se obtengan datos sensados habrá que realizar conversiones espaciales para determinar a qué posición del mapa corresponden los mismos pero, esta vez, el costo de las conversiones depende de la cantidad de datos de sensado que produzca el robot. Ya que la arquitectura está orientada a robots autónomos se supone que dispondrán de abundantes datos a procesar -debido a que se necesitan múltiples sensores para desambiguar algunas situaciones- de ahí, que no se considere tampoco ideal este esquema. Más aún, los datos de sensado suelen ser efímeros ya que los presentes son muy parecidos a los previos y en particular los datos de entidades que estén muy próximas al robot pueden generar un flujo de datos importantes que cuando el robot se aleje de las mismas sólo importará de aquellos datos -en la mayoría de los casos- la posición e identificación de la entidad. De ahí que convenga no convertir los datos del sensado en datos topo referenciados hasta que no sea el momento de alejarse de la entidad sensada.

Es por estos inconvenientes que se presentan (abundantes datos de sensado espacialmente referenciados y que el modelo contenga muchos hitos) que se propone utilizar un sistema híbrido ego - topo centrado relativo.

Sistema híbrido relativo

El sistema de representación elegido es uno relativo por lo cual las entidades toman como punto de referencia espacial a alguna otra entidad no siendo está última necesariamente la misma para todas. Inclusive, es posible tener una misma entidad con varias entidades de referencia cuando la misma es usada por varias fuentes. El carácter híbrido se denota ya que uno de los puntos de referencia espacial relativos debe ser siempre la entidad *Self* -que como se vio en la introducción de este trabajo representa al propio robot dentro del modelo- de modo que se gestiona tanto la forma egocéntrica como la topológica estableciéndose un único punto egocéntrico y varios puntos para el esquema topológico. Por lo tanto, todos los puntos de referencia son entidades del modelo pero no todas las entidades son puntos de referencia y en particular la entidad

Self estará siempre configurada como punto de referencia para cumplir la función de la referencia egocéntrica. Además, *Self* es la entidad central del escenario y es también a la cual todas las demás entidades del mismo referencian. La vinculación de *Self* con el mapa se realiza con una o más relaciones espaciales a entidades *mojones* del mapa que se verá más adelante.
La validez del esquema depende de que los nodos dispongan de una orientación destacada. Para esta orientación se considera el sistema de coordenadas locales sobre el cual se construyen los prototipos de las entidades -ver sistema de modelización de objetos- constituyéndose así este sistema de referencia intrínseco de las entidades en el cimiento para las relaciones espaciales del modelo.
Cómo se citó al comienzo de la sección las entidades son el punto de acceso al modelo y son en las instantáneas en dónde se registran la mayoría de la información sobre las entidades y, es por eso que para describir precisamente el modelo, es necesario aclarar que las relaciones espaciales se plantean entre las instantáneas de las entidades. Más adelante cuando se traten las particularidades del mapa y del escenario se volverá sobre las relaciones espaciales.

Relación espacial

Se define la relación espacial entre dos instantáneas de entidades distintas A y B que no sean de tipo abstracto -ver sistema de modelización de objetos- a la posición de la instantánea B con respecto al sistema interno de referencia que utiliza el prototipo de la entidad de la instantánea A. Puesto que las relaciones son simétricas también se expresa la posición de la instantánea A con respecto al sistema interno de la instantánea B. Se completa la definición estableciendo que toda instantánea no abstracta debe tener al menos una referencia espacial con una instantánea de otra entidad. Las instantáneas abstractas carecen de relaciones espaciales.
La descripción de la relación espacial considera las seis dimensiones espaciales y además las seis tolerancias asociadas a las mismas.

Atributo movilidad de la relación espacial

Dado que el mapa constituye un grafo de instantáneas de entidades que se referencian las unas a las otras es muy conveniente que las relaciones entre las instantáneas tengan una propiedad que señale si los valores de la relación espacial entre dos instantáneas son estáticos –es decir no cambiarán en el futuro- o por lo contrario que sea altamente probable que cambien indicando que es una relación espacial dinámica. De este modo, en aquellos casos que es posible, se podría seleccionar como entidades de referencia aquellas que unen sus instantáneas a través de una relación espacial estática con la finalidad de realizar menor cantidad de ajustes de posición.
Principalmente existen dos formas para determinar el carácter de movimiento de una relación espacial. El primero, experimentalmente, por medio de mediciones recurrentes de los ajustes de posición y, el segundo, por información sobre el dominio de aplicación que el robot tenga incorporado en las evoluciones especulativas de las entidades. No obstante, ninguna de las dos formas de identificar el carácter móvil de una relación espacial es excluyente del otro aunque, habrá que realizar algunos ajustes finos, como por ejemplo si desde el dominio de aplicación se determina que una relación es estática y desde el sensado se percibe una breve fluctuación de la posición entonces habría que considerar que esto último sólo se debe a errores del sensado.

Relación de referencia

Una relación de referencia es una relación espacial en la cual uno de los extremos es llamado referente y el otro referenciante. Además el extremo referente deber ser una instantánea descriptiva o especulativa.
La referencia puede realizarse al centro de la entidad o al centro de un *elemento* de la entidad -ver sistema de modelización de objetos-. El sentido de utilizar los elementos es que cuando las entidades referenciante y referente estén muy próximas o una entidad sea muy grande con respecto a otra la referencia al centro pierde sentido práctico. Por ejemplo, si la entidad referenciada es una pared larga y el robot está

cerca de la pared una referencia al centro de la entidad pared puede ser muy poco práctica para un desplazamiento bordeando la pared. En cambio, si suponemos que una pared está constituida por muchos elementos entonces se puede hacer la referencia al elemento más cercano y de esa forma hay una referencia que es casi ortogonal a la pared que es más práctica para un seguimiento de pared.

Representación temporal

Como ya se comentó en “Instantáneas pasadas, presentes y futuras“ -pag. 36- el orden temporal surge de las restricciones de ejecución que vinculan a las instantáneas. Aquí se señala los aspectos de implementación definiéndose que cada instantánea sea un nodo y las restricciones de ejecución un conjunto de arcos dirigidos. De esto se desprende que cuando se lleve a cabo una sincronización de distintas fuentes -ver sistema de materialización del lenguaje- se conformará un grafo dirigido acíclico.
Si las instantáneas que se registran en el escenario se limitan al presente se tiene un robot reactivo y, si se considera el futuro, agregando instantáneas siguientes a la del presente, se contará con una breve planificación en la que se puede controlar su alcance gradualmente determinando cuántas instantáneas se consideren. Lo interesante de esta concepción, es que se puede ajustar dinámicamente la planificación-reacción según las circunstancias.

Instantánea presente

Explícitamente se define para cada nodo entidad cual es su instantánea presente que puede ser especulativa, descriptiva sensada o descriptiva estimada -ver próximo párrafo- y de ahí, por navegación sobre los arcos de sincronización del grafo, se puede obtener su pasado y/o su futuro.

Instantáneas descriptivas sensadas, estimadas y predichas

El ciclo de vida de las descriptivas es distinto si la misma proviene del SP o del SML. Las del SP siempre son el presente y reemplazan a la instantánea presente descriptiva o especulativa asociada a la entidad independientemente si su origen sea del SP o SML pasando la reemplazada a conformar el pasado más inmediato si el modelo utiliza el pasado. De otro modo, se descarta.
Para el caso de las instantáneas descriptivas que el SML genera por medio del recorrido de una evolución asociada al prototipo de la entidad se denominan estimadas sin son para el presente y si son para el futuro predichas.
Las estimadas se construyen a partir de información del pasado sensada -i.e. una instantánea descriptiva del SP- para determinar el presente; en cambio, las predichas generan instantáneas descriptivas para el futuro en función de la información del presente independientemente si el origen de la misma sea el sensado o una estimación.
El concepto fuerte con las descriptivas originadas en el SML es que necesitan un dato sensado cercano en el tiempo para que, a través de los algoritmos de la evolución, generar las instantáneas descriptivas y, sólo una para cada instante. La primera descriptiva de futuro -las que predicen- se ponen a continuación, mediante una restricción de ejecución, de la descriptiva presente. Con las siguientes descriptivas de futuro se opera análogamente pero trabajando sobre la descriptiva de futuro anterior inmediata.

Instantáneas especulativas

Las instantáneas especulativas pueden ser para el pasado, presente o futuro y sólo son válidas si no existen descriptivas que cubran el mismo evento. A diferencia de las evoluciones descriptivas, las especulativas no hacen uso necesariamente de la información sensada de la entidad para la cual generan instantáneas. Puesto que pueden, utilizan además información que tengan sobre el dominio de aplicación del robot y sobre la posición de otras entidades para especular la posición dónde podría estar la entidad en cuestión.

Instantánea especulativa principal

La otra característica distintiva de las evoluciones especulativas con respecto a las descriptivas es que pueden generar, para un mismo período de tiempo, más de una especulativa para cubrir distintas posibilidades. Sin embargo, siempre una de las instantáneas que cubren distintas posibilidades para un mismo hecho es marcada como *principal* para facilitar en algunos contextos de uso su utilización. Si bien no es requisito de la arquitectura generalmente se selecciona como instantánea principal a aquella que la evolución considere más probable de que sea cierta.

Instantáneas prescriptivas

En el grafo las instantáneas prescriptivas declarativas tienen un ciclo de vida que comienza cuando son generadas por las fuentes y termina cuando llegan al presente o son descartadas, de lo cual se deduce que las prescriptivas jamás forman parte del pasado. Durante el ciclo vida de las instantáneas sus fuentes verifican que las instantáneas que han generado se ajustan a la información más actual y, de haber diferencias, van modificándolas para adecuarlas a los datos más actualizados. No obstante, algunas veces las instantáneas son eliminadas por sus fuentes ya que determinaron que ese futuro ya no es posible o necesitado.

La vinculación con la información más actual queda reflejada en instantáneas descriptivas o especulativas pues toda instantánea prescriptiva debe tener como antecedente inmediato a una descriptiva o especulativa. Las instantáneas prescriptivas se vinculan entre sí por medio de las restricciones de ejecución para representar el orden de ejecución necesitado.

Relaciones espacio temporal del futuro

Las instantáneas descriptivas o especulativas del futuro mantienen las mismas relaciones espaciales que la instantánea presente que le corresponde. El criterio para mantener las mismas relaciones espaciales es que como el futuro es muy próximo la diferencia de posiciones no pueden ser muy grandes evitándose así tener que buscar otras entidades para plantear las relaciones espaciales. Los valores de las relaciones espaciales se establecen al ajustar las posiciones de acuerdo a aquellas instantáneas de la primera entidad que se interseccinan en su dimensión temporal con las instantáneas de la segunda entidad.

Si hubiera más de una en la intersección entonces hay que plantear una relación espacial con cada una de estas ya que probablemente impliquen una diferencia en la posición espacial. En particular hay que satisfacer esto para las instantáneas de los extremos temporales que sus rangos de la dimensión temporal son abiertos. Así, si una entidad tiene instantáneas futuras y está relacionada con otra entidad que aún no tiene instantáneas de futuro para aquel momento entonces se relaciona con la última instantánea ya que como tiene un rango abierto intersecciona con la primera. Ante el supuesto caso que empiecen a agregarse instantáneas de futuro a continuación cada relación que se vincule con un rango abierto es analizada nuevamente para observar si hay que cambiar las relaciones o si continúan en la última del rango abierto. Recíprocamente si instantáneas de futuro dejan de existir todas las relaciones espaciales son movidas a la última instantánea que quede para la entidad.

Escenario

El escenario es la parte del modelo del SIET en donde ocurre la acción. Por eso es que puede decirse que el escenario se va desplazando a través del mapa y que *Self* es la entidad de referencia que vincula todas las entidades del mismo ya que cuando aquellas entran dentro del escenario son automáticamente referenciadas con *Self*.

Cuatro parámetros determinan el alcance espacial y temporal del escenario. El primero es el parámetro *amplitud espacial* cuyo valor determina el radio de una esfera con origen en *Self* que fija el tamaño del escenario para las entidades que no están siendo sensadas. Los dos siguientes parámetros se refieren a lo temporal, es decir, a la dimensión temporal -atributo *time*-. El primero es el parámetro *persistencia de memoria de trabajo* que indica durante cuando tiempo serán registradas las

instantáneas más allá de haber alcanzado el presente y el segundo parámetro es el *horizonte de planificación* que determina en general -ya que también está el umbral de planificación- hasta cuando en el futuro se podrán generar instantáneas.
Finalmente el parámetro *contexto de acción* determina una esfera menor a *amplitud espacial* que indica las entidades que el sistema de acción procesará en su motor de dinámica. Es el subconjunto de las entidades del escenario que están tan cerca del robot que tienen que considerarse en sus movimientos.

Mapa

El mapa es básicamente el registro atemporal de las instantáneas descriptivas o especulativas de las entidades que estuvieron activas durante la existencia del robot. Esto último expresa el hecho de que el mapa persiste a los reinicios del robot. Es atemporal ya que la dimensión temporal sólo se mantiene en el presente como una formalidad para poder ser compatible con el escenario. La persistencia del mapa tras el reinicio se sostiene no sólo para evitar construir desde cero todo el grafo del mapa ante cada reinicialización sino principalmente bajo la presunción que el robot no se desplazará demasiado desde la última posición conocida. Se espera que esto permitirá al robot volver a orientarse más rápido. Más adelante se trata específicamente el tema de la orientación del robot.
El mapa se puebla cuando las entidades abandonan el escenario o cuando una entidad activa de una fuente no entró dentro del escenario. También más adelante se verá como se mantiene el mapa.
Un tema cercano al mapa es la forma en cómo se podría representar la información espacial que contiene en robot acerca del dominio de aplicación desde el diseño. La forma propuesta es mediante evoluciones especulativas de forma de conformar mapas de “parches” alrededor de una entidad.
La idea es construir una evolución especulativa que cree tantos recorridos especulativos como entidades haya en el “parche” de forma que todas las evoluciones involucradas tengan la misma entidad de referencia –el centro del parche-. Cada recorrido tomará a su cargo generar una instantánea especulativa para una de las entidades del parche y la relacionará con una alta calidad con la entidad de referencia. Cuando se invoque la evolución que construye el parche se crearán todas las entidades -si no estaban previamente creadas- y se relacionarán espacialmente entre sí de acuerdo a lo establecido por el “parche”. Luego los recorridos se auto apagarán pero en el mapa quedarán las entidades y las relaciones espaciales hasta que otro recorrido las cambie o pase el escenario por arriba y las sense y cambie las instantáneas especulativas por descriptivas.

Inicialización

La inicialización del robot implica también la inicialización del SIET. El proceso comienza con la creación del escenario con la entidad *Self* dependiente de la fuente *Main*. Con esa configuración sólo se podrán expresar movimientos para las entidades propias con respecto a Self pero no se podrá actuar sobre el resto del mundo por eso es necesario obtener desde los parámetros de reinicio del robot cual es la *presunción de posición de partida*.
Si la presunción de partida es estructurada, esto es, el robot parte de una posición conocida de antemano, habrá una evolución especulativa que determine con una alta confiabilidad la posición del robot con respecto a otras entidades[3]. Así cuando *Main* active la antedicha evolución se creará el recorrido correspondiente el cual entonces hará pedidos de entidades al SIET. Si las entidades ya están en el mapa -i.e. no es el primer reinicio del robot- se vincula a *Self* con aquellas con una alta confiabilidad de acuerdo a lo establecido por el recorrido especulativo. Por otro lado, si las entidades no están presentes en el mapa se creerán en el mapa o en el escenario según a la distancia que estén de *Self* y luego se vincularán como en el caso anterior.

[3] En una aplicación muy pequeña se pueden declarar las entidades directamente en el código.

La otra posibilidad para la *presunción de posición de partida* es que deba ser hallada autónomamente. En este caso, también hay dos posibilidades, que haya un mapa preexistente -i.e. no es el primer reinicio del robot- o que no lo haya. En el segundo caso, *Main* activará alguna evolución de aplicación y sus recorridos derivados comenzarán a crear entidades y así la primera que abandone el escenario comenzará a construir el mapa y a establecer una posición de *Self* con respecto a aquel. Si el robot se reinicia antes de que alguna salga del escenario no se construirá el mapa y el próximo reinicio también se partirá sin mapa.
En cambio, para el primer caso, el SIET establece una vinculación de *Self* con las entidades de la última posición conocida pero ajusta la confiabilidad de las dimensiones espaciales en valores mínimos de modo que se "comience" por la última posición conocida pero dejando un margen de incertidumbre para descartar aquella vieja posición si los datos que provengan de los sensores no son consistentes con aquella.
Aquí el SIET se detiene a esperar las solicitudes de pedidos y/o las instantáneas generadas por los sistemas con los cuales se vincula.

Gestión del mapa

El mapa tiene entidades concretas y virtuales con una única instantánea atemporal -tiene ambos extremos del período abiertos- que puede ser descriptiva o especulativa y casi con seguridad relaciones espaciales a las instantáneas atemporales de las entidades cercanas que se encuentren en las regiones que surgen de dividir la dimensión rumbo y altura más o menos del mismo tamaño angular (como una esfera geodésica de hexágonos y pentágonos regulares).
Existen dos maneras para crear una nueva entidad en el mapa. La primera es que un recorrido active una entidad cuya ubicación esté más allá del límite del escenario y que además no se precise sensar ya que si no correspondería estar en el escenario. A continuación se crea una única instantánea atemporal hasta que empiece a sensarse o la entidad entre dentro del alcance del escenario. Esto es así porque como está tan lejos no tiene sentido incluirla en los planes de acción que son muy breves ni tampoco hará uso de los recursos -sensores y actuadores- del robot. Igualmente el recorrido que la incluyó puede actualizar su posición pero no habrá registro temporal de ello. Si la entidad de referencia que utilice el recorrido no está en el modelo y deba ser creada, entonces se creará una componente no conexa en el modelo hasta que el escenario pase por ahí y logre conectarla al resto del modelo u otro recorrido vincule la entidad de referencia con el resto del modelo.
La otra manera de que se registre una nueva entidad es cuando la misma fue creada en el escenario e independientemente si fue el escenario el que se desplazó o que lo haya sido la entidad, si la misma quedó afuera del escenario pasará a ser parte del mapa. En ese momento dejará de ser *Self* su entidad de referencia así como también se eliminará la relación espacial con aquella y a cambio se buscarán entidades en el mapa con la cual relacionarlas espacialmente. En este caso si la entidad saliente no es usada más por ninguna fuente entonces no tendrá entidad de referencia.
Para sacar una entidad del mapa hay también dos posibilidades. La primera que lo haga el procesamiento de la actualización del escenario ya que cuando una entidad del mapa entra en el escenario quizás después no vuelva al mapa. Esto se verá más adelante cuando se trate el escenario.
La segunda que un recorrido decida eliminarla pero en ese caso la entidad no tendría que haber sido sensada. Esto es especialmente útil para las entidades virtuales que pueden estar asociadas con un proceso en particular y luego del mismo pierden todo sentido de ser y también es práctico para eliminar entidades especulativas sin que haya aparecido una descriptiva. Cuando las entidades se eliminan, se revisan si las mismas conforman puentes para vincular espacialmente a otras entidades; si es así, entonces se replantean esas relaciones espaciales para omitir la entidad a ser eliminada y luego se elimina.

La gestión de las relaciones espaciales tiene que mantener bien conectado al mapa. Así cada vez que un recorrido actualiza la posición de una entidad del mapa se revisan las relaciones espaciales de modo que las mismas se distribuyan uniforme a lo largo de la dimensiones rumbo y altura conectándose con las más cercanas. Se propone para la actualización analizar las entidades alcanzadas hasta con dos saltos a partir del par surgido de la relación espacial que se modificó de modo que localmente se vaya construyendo un grafo del mapa bien conectado. A partir de ese análisis se eliminarán, agregaran o actualizarán las valores de las relaciones espaciales. De esa forma al "desplazarse" el robot por el mapa siempre habrá una entidad que antecede a otra de modo que sólo se detecten por el sensado las entidades que son "nuevas" en la ubicación en cuestión y no las "viejas" que no fueron consideradas porque la exploración en el grafo del mapa no alcanzó a cubrir los nodos que las representaban.

Acceso al modelo

Como se ha visto el modelo del SIET consta de un escenario y de un mapa y que dichas partes contienen distintos tipos de información pero que comparten las entidades y las relaciones espaciales. Precisamente sobre esos elementos compartidos es que se establecen los mecanismos de acceso al modelo. Básicamente todas las entidades son indexadas por prototipo y por identificador de modo que, por ejemplo, con un acceso a través de un árbol de búsqueda se puede determinar la existencia de una entidad o de un prototipo en tiempos logarítmicos.

Para cada componente conexa del modelo además se construye también un árbol espacial[4] de modo que con tiempos logarítmicos se puede obtener la posición relativa entre dos par de entidades -si es determinable, esto es, que pertenezcan ambas a la misma componente conexa-. Conocidas ambas posiciones se agrega una relación espacial siempre y cuando una de las entidades no sea *Self*. Caso contrario, no funcionaría el mecanismo de control que verifica que el grafo del modelo no crezca en demasía.

El mecanismo de control de las relaciones espaciales consiste en sumar un punto cuando pasan ambos extremos del arco de la relación espacial por el escenario y pierden un punto cuando un extremo entra y sale del escenario sin que el otro extremo entrara en escena. De esta forma entidades cercanas se mantendrán espacialmente vinculadas. Una implementación sencilla por ejemplo consistiría en utilizar un número entero y que cuando llegué al máximo positivo se omita de seguir incrementando pero cuando se llegue al máximo negativo entonces que se elimine la relación espacial.

Gestión del escenario

Orientación

Se define que *Self* está orientado si las entidades sensadas coinciden con las entidades del modelo -mapa y escenario- tanto en la identificación como en la posición de las mismas.

Incorporación de entidades al escenario

Un entidad es incluida en el escenario si es una entidad activa -esto es, aquellas que están siendo usadas efectivamente por las fuentes y no que las entidades estén meramente declaradas- y que además esté siendo sensada o que una instantánea de esta se encuentre a una distancia de *Self* menor que el valor del parámetro *amplitud espacial*.

Para poder determinarse la distancia a *Self* de una instantánea de una entidad la misma debe estar en la componente conexa en la cual *Self* está orientada. Si no es así, aquellas entidades no serán agregadas al escenario sino que serán tomadas por el mapa. Es por eso que en la práctica el escenario comienza a construirse a partir de entidades sensadas, luego se orienta con respecto al mapa –ya sea porque reconoció una escena en el mapa (en breve tratado), porque empezó a construir un mapa a

[4] Por ejemplo Quadtree, R-tree, kd-tree u otros similares.

medida que se desplaza o porque se partió de una posición de arranque conocida de antemano- y finalmente en ese momento puede empezar a agregar entidades al escenario que no estén siendo sensadas.
Puesto que las entidades del escenario están vinculadas siempre con *Self* y se usan a través de esta es posible -para ahorrar cómputo- que sus relaciones espaciales del mapa dejen de actualizarse -a excepción de aquellas que se vinculan con *Self*- hasta que abandonen el escenario y entonces sí se actualizará el valor de las relaciones con el mapa.

Exclusión de entidades del escenario

Movilidad de las relaciones espaciales

En el momento en que las entidades salen del escenario es cuando se evalúan las relaciones espaciales de movilidad para determinar experimentalmente si son estáticas o dinámicas.

Entidad mojón

También en ese momento se clasifican las entidades vinculadas con relaciones espaciales estáticas con el propósito de establecer cuales grupos de entidades conforman *mojones*. Los mojones son entidades virtuales que se relacionan con el grupo de entidades que no se mueven entre sí. *Self* está orientado si está conectado con uno o más mojones.

Reconocimiento de escena

Una de las funciones más importantes de la gestión del escenario es reconocer si la información proveniente del sensado es consistente con el modelo del escenario.
Lo que se busca es emparejar tanto la identificación de las entidades como las posiciones relativas entre ellas. El reconocimiento tiene que hacerse en cada una de las componentes conexas del modelo ya que puede haber intersección de entidades. De hecho esta intersección es la base para la unificación de las componentes conexas.
El reconocimiento de la escena es necesario para determinar la orientación del modelo y limpieza del modelo como se verá más adelante.
El reconocimiento puede ser pasivo si solamente el SIET aprovecha las entidades que el sistema de percepción le entrega o activo si existe un recorrido que solicite el sensado por un breve tiempo de las entidades que estén en el escenario pero que no estuvieran siendo utilizadas para algún otro fin. Ver más adelante en “Reconocimiento activo” en la página 61.

Orientación y limpieza del modelo

El reconocimiento de la escena es vital para las funciones de orientación y limpieza del modelo. Si el factor de coincidencia entre el modelo y lo sensado es alto -es decir, tanto la identidad de las entidades como sus posiciones coinciden entre el modelo y lo sensado- determina que se declare al robot como orientado registrándose la posición de *Self* en el mapa con respecto a algunas entidades mojones. Si el factor de coincidencia es medio y puede ser aumentado eliminando una entidad del escenario que no tiene ninguna correlación con el sensado entonces se elimina la entidad del escenario y por ende también del mapa cuando el escenario abandone esa zona del mapa cayendo nuevamente en el paso anterior. Si el factor de coincidencia es bajo o en todo caso es medio pero no se puede corregir eliminando una entidad con los restricciones anteriores entonces se declara como desorientado al robot. Finalmente mientras el robot se mantenga orientado se actualizará en el mapa cada tanto tiempo la posición de *Self* con respecto a algunas entidades mojones por si se produce un reinicio del robot.

Actualización de relaciones espaciales

La utilización de un modelo relativo para expresar las posiciones de las entidades determina que cuando una entidad se mueva haya que actualizar las referencias espaciales que la conectan con otras entidades. Una entidad se mueve cuando una

fuente informa al SIET que generó una nueva instantánea y, como ya se mencionó, se expresa con respecto a una instantánea descriptiva o especulativa de la entidad de referencia. Una instantánea puede tener varias relaciones espaciales que son gestionadas automáticamente por el SIET para mantener conectado al grafo pero tiene una única relación espacial que es determinada por su fuente y esa es precisamente la relación espacial con su entidad de referencia. Se presentan tres casos posibles de generación de instantáneas que se verán a continuación: generada por el SML, por el SA o por el SP.

Generada por el SML

La instantánea es expresada con respecto a una instantánea de la entidad de referencia y así se registra directamente en el modelo. El único impacto posible son las relaciones espaciales que la entidad de la instantánea generada pudiera tener más allá de la relación que tiene con la entidad de referencia. En ese caso, se considerará que aquellas entidades no se mueven y se agrega una relación espacial que una la instantánea generada con la instantánea de las otras entidades que le corresponda por la intersección temporal. La anterior relación espacial que hubiera quedará en el pasado o se eliminará.

Después cuando se actualicen las demás entidades se agregarán nuevas instantáneas y posiblemente estos últimos arcos agregados a la instantánea de extremo abierto sean movidas a la nueva instantánea. Ver -Relaciones espacio temporal del futuro pag. 42-.

Generada por el SA

Las instantáneas prescriptivas que genere el SA siempre serán sobre entidades propias de tipo actuador. Todas serán generadas con respecto a *Self* y por lo tanto se está en un caso particular de las generadas por el SML en donde la entidad de referencia es *Self*.

Generada por el SP

Para el caso de las instantáneas generadas por el SP se agrupan según sea la entidad de la instantánea propia o no. Para el primer caso se está en un situación idéntica a las generadas por el SA, en cambio, para las no propias se requiere de un procesamiento más elaborado.

Todas las instantáneas no propias generadas son descriptivas de presente y su entidad de referencia es siempre *Self*. Aquí se presentan cinco casos posibles:

- Se modificó la relación espacial
  a. Se movió la entidad sensada pero no *Self*.
  b. Se movió *Self* pero no la sensada.
  c. Se movieron la sensada y *Self*.
- No se modificó la relación espacial
  d. Ni *Self* ni la entidad sensada se movieron.
  e. *Self* y la entidad sensada se movieron, pero lo hicieron al unísono con respecto a otro sistema de referencia.

El propósito con las instantáneas generadas por el SP es identificar las entidades que se hayan movido para poder llevar a cabo la actualización de las relaciones espaciales con un tratamiento igual al utilizado para las generadas por el SML. Sin embargo, es necesario antes plantear un marco que posibilite determinar si una entidad se movió o no ya que, como se observa de los cinco casos citados, no se desprende directamente.

Perímetro: Se define como *perímetro* a las entidades que rodean a *Self* tales que o bien pertenezcan al escenario y no estén siendo sensadas o bien pertenezcan al mapa y, que en ambos casos, estén relacionadas espacial y directamente con una entidad del escenario que esté siendo sensada.

Para comenzar se establece que todas las relaciones espaciales que vinculan a entidades del escenario que están siendo sensadas con las pertenecientes al perímetro se les suspenda la actualización espacial porque, de hecho, estas serán las relaciones espaciales que podrían ser actualizadas por el desplazamiento de una entidad que esté siendo sensada.
Se considera que las relaciones espaciales entre *Self* y las entidades sensadas del escenario son correctas porque es el sistema de percepción que se encarga de actualizarlas pero el problema es que no existe un marco absoluto en el cual sostenerse para determinar cuáles entidades se movieron y cuáles no. Si *Self* estuviera "fijo" sencillamente las entidades sensadas que varían su relación espacial con el momento anterior serían las que se habrían movido y de esa forma todas las relaciones con entidades del perímetro deberían actualizar sus relaciones espaciales tal como se hizo con las instantáneas generadas por el SML.
Entonces el problema se reduce a identificar cuándo y cómo *Self* se mueve con respecto a las entidades sensadas para así poder compensar el movimiento de *Self* y determinar cuáles entidades se han movido además de *Self*.

La idea del proceso es tener en todo momento dos muestras consecutivas obtenidas por sensado de las entidades que estén siendo sensadas separadas por un lapso de tiempo tan breve que permita que *Self* sólo se haya podido mover muy poco. También se define para cada dimensión de la posición de *Self* un delta pequeño. Así a partir del primer conjunto de muestras se considera cual es la posición de origen de *Self* y luego cuando se reciben el segundo conjunto de muestras se realiza una comparación del siguiente modo:

1. Para cada dimensión de la posición de origen de *Self* se calculan tres valores que resultan de sumar, restar y no aplicar el delta correspondiente a la dimensión en cuestión.
2. Se calculan todas las combinaciones posibles sobre la posición de origen de Self. Esto es igual a $3^{\text{cant. de dimensiones}}$ que en caso de ser seis da 729 que representa todos los posibles movimientos pequeños que podría haber hecho *Self*.
3. Se calcula para cada movimiento posible de *Self* qué posición tendría que tener cada entidad sensada en su posición del primer conjunto desde esa perspectiva -i.e. se presume que no se movió la entidad sensada-.
4. Se compara la posición sensada del segundo conjunto de cada entidad con la obtenida en el paso anterior.
5. Se determina que el movimiento que hizo el robot es aquel con mayor cantidad de aciertos obtuvo en el paso anterior.
6. Se determina que las entidades que se movieron fueron aquellas que no coincidía la posición calculada sobre los datos del primer conjunto con los datos sensados del segundo conjunto.
7. Se seleccionan las entidades que se movieron y que además pertenezcan al "perímetro" para que se les actualicen las relaciones espaciales tal como se realiza en el caso del SML.
8. El segundo conjunto de muestras se convierte en el de origen y con el arribo del tercer conjunto de datos sensado se repite el ciclo.

Cabe aclarar que el esquema propuesto funciona sin considerar información de navegación inercial, odometría o de balizas externas; sin embrago, en caso de que tal tipo de información estuviera disponible la misma es perfectamente integrable al esquema que se propone haciendo que sea más sencillo alcanzar el resultado buscado al determinar directamente cual es la posición de *Self* a verificar. También se nota que el problema es práctico para resolver paralelamente ya que puede procesarse independientemente cada movimiento tentativo.

Grafo del modelo

En el presente título se describen los distintos tipos de nodos y de arcos presentes en el modelo para implementar el mapa y el escenario de modo que después se pueda describir cómo funciona el mismo.

Los nodos representan:

- **[fuente]**: son los nodos representantes de los procesos de los sistemas de motorización de lenguaje, percepción y actuación.
- **[entidad]**: incluye también a las subentidades como entidades.
- **[requerimiento]**: en los tres propósitos de uso.
- **[instantánea]**: en los tres propósitos de uso.
- **[configuración de hardware]**: para cada entidad recurso -sensor y actuador- hay un conjunto de configuraciones de hardware que ajustan los parámetros de control del recurso -ver sistema de hardware- y para cada configuración solicitada es representada con uno de estos nodos.
- **[orden en cola]**: es un nodo que representa la utilización del recurso en una configuración de hardware dada por un período de tiempo. Estos nodos se agregan en forma de cola tras cada nodo de configuración de hardware. Sirven para que las fuentes del SP y del SA puedan planificar el uso intercalado de los recursos a la vez que es soporte para la verificación de deadlocks.

Los arcos representan relaciones:

- **[espacial]**: relación espacial entre un par de nodos [instantánea]. Es simétrica.
- **[referencia]**: señala la entidad de referencia de una fuente. Une una [fuente] con una [entidad].
- **[necesidad]**: indica que una [fuente] necesita a una [entidad].
- **[solicitud de requerimiento]**: une una [entidad] con un [requerimiento].
- **[importancia]**: une una [fuente] con un [requerimiento].
- **[sincronización]**: declara la existencia de una restricción de ejecución. Es entre un par de nodos [instantánea]. La navegación por esta relación es de ida y vuelta.
- **[instantánea presente]**: entre un [requerimiento] y una [instantánea] descriptiva o especulativa.
- **[espera de recurso]**: se utiliza entre los nodos [orden en cola] para vincular entre sí este tipo de nodo. La vinculación puede ser entre nodos de una misma configuración, distintas configuraciones de una misma entidad recurso y entre distintas entidades recurso. Pero para todos los nodos excepto los últimos de las colas tienen que tener uno y sólo un arco entrante y todos tienen que tener un único arco saliente.
- **[es configuración de]**: une una [entidad] recurso -sensor o actuador- con las [configuración de hardware] que tengan colas activas.
- **[es cola de]**: une una [configuración de hardware] con el nodo [orden en cola] cabecera de su cola asociada.
- **[asignación]**: entre el primer nodo [orden en cola] de una [configuración de hardware] y un [requerimiento] tal que el nodo [orden en cola] no tenga predecesor por la relación [espera de recurso].
- **[uso pasivo]**: entre nodo [orden en cola] que tenga una relación de [asignación] y un [requerimiento]. No tiene costo pero tampoco garantías ya que utiliza un recurso que puede ser usado de un momento a otro por otro requerimiento.
- **[pedido de recurso]**: entre una [instantánea] de futuro de una entidad propia y un nodo [orden en cola] distinto al que tenga una relación de [asignación].

- **[materialización]**: vincula un [requerimiento] objetivo y una [fuente] coadyuvante -ver Fuente coadyuvante Pág. 55-.

Carácter híbrido de la arquitectura

Si bien un modelo, un mapa y un plan puede llevar a pensar en una arquitectura del tipo deliberativo, la que aquí se presenta no lo es, y el modo en cómo se gestiona la dupla mapa – escenario y el balance entre reacción y deliberación es lo que permite que sea denominada como híbrida.

El primer rasgo distintivo es que el escenario es mantenido en el menor tamaño posible con el propósito de que el tiempo necesario para su gestión quepa dentro de los márgenes del tiempo real del robot. El criterio que se usa para definir el tamaño del mismo es la cantidad de entidades representadas y para acotar precisamente ese número es que sólo se registran las entidades que están siendo utilizadas en cada momento por las fuentes. Así, el escenario sólo contiene las entidades sobre las cuales el robot puede "reaccionar".

Justamente para cumplir con ese propósito en el escenario además de las entidades en uso y sus instantáneas también se guarda un identificador para representar a cada fuente. La idea es que las entidades estén asociadas a las fuentes que las usan de modo que, cuando una fuente termine, se desvincule de las entidades que estaba usando. Es ahí que si la entidad queda sin ninguna fuente asociada entonces el SIET la elimina del escenario. A su vez, el SIET antes de crear una nueva entidad a pedido de las fuentes buscan en el modelo -mapa y escenario- si la misma ya no estaba creada con el fin de poder reutilizarla. No obstante, esto, más que una cuestión operativa para acotar el crecimiento del escenario, se relaciona con el hecho que la fuente puede necesitar una entidad en particular; y así, este tema, está del lado de lo que significa la entidad para la aplicación y se trata en la sección del sistema de motorización del lenguaje (SML).

Hitos de interrupción

El otro rasgo distintivo está relacionado con el manejo del tiempo y con la planificación ya que esta es dinámicamente ajustable de modo que el robot puede ser totalmente reactivo, si sólo contiene en el escenario instantáneas presentes, o puede tener una breve planificación -apenas unos segundos como mucho- para intentar realizar una mejor asignación de recursos de la que surgiría de una configuración puramente reactiva. Para llevar a cabo esta integración cuando se planifica se puede marcar ciertas instantáneas del plan como hitos de interrupción. Así, una instantánea marcada como hito de interrupción significa que ese plan se puede interrumpir en ese momento para después poder continuarlo. Por ejemplo, si el plan establece subir un objeto con un brazo desde el piso hasta apoyarlo sobre una mesa y después que deba empujarla entonces la instantánea que marca que el objeto se apoyó sobre la mesa será el hito de interrupción. En ese punto el plan podría ser interrumpido por ejemplo para utilizar el brazo para otra necesidad.

Descripción general de servicios

El SIET en su funcionamiento se comunica e integra la información proveniente de los sistemas de motorización del lenguaje -SML-, de percepción -SP- y de actuación -SA- devolviendo a cada uno de estos una visión integrada y sintética del mundo. Esa visión es entregada por el SIET a sus proveedores de datos cuando lleva a cabo los siguientes servicios:

- Brindar información descriptiva y especulativa sobre las entidades
- Seleccionar las instantáneas procedimentales a ejecutar
- Ofrecer una memoria espacio temporal para las entidades
- Presentar su información de acuerdo a un marco de referencia solicitado
- Resolver consultas sobre relaciones espacio-temporales entre entidades

- Determinar las entidades próximas del robot
- Establecer un umbral de planificación

los cuales son descriptos uno a uno a continuación.

Brindar información descriptiva y especulativa sobre las entidades

Este servicio comienza recibiendo un requerimiento que será analizado por el SIET de acuerdo al propósito de uso del requerimiento y la dimensión *time* del mismo. Al inicio, el proceso de la respuesta, en todos los casos, se comportan en forma similar. Así, como primer paso se busca en el modelo una instantánea que sea del propósito de uso requerido para la entidad y que además satisfaga el requerimiento de la dimensión temporal. Si se halla la instantánea, entonces se verifica si cumple con los demás requerimientos -confiabilidad y precisión en las otras dimensiones- y, si los cumple, se devuelve esa instantánea.
Sin embargo, si no se encuentra la información en el modelo entonces el SIET se ocupará de obtenerla, de registrarla en el modelo y luego con esta responderá la solicitud. Según el caso, la información faltante se obtendrá con un llamado al SP o con la creación de un recorrido específico para la entidad en el SML. Los requerimientos recibidos para las instantáneas de las entidades persisten en el SIET hasta que la fuente que lo envió decide cancelarlo; sin embargo, cuando una fuente termina se cancelan todos sus requerimientos directamente. A continuación se presentan las alternativas existentes en la resolución de los requerimientos recibidos por el SIET.

- Información del pasado
  - Se busca en el escenario una instantánea descriptiva del pasado para la entidad que cumpla con el requerimiento de la dimensión *time*.
    - Si se halla, se verifica si cumple con los demás requerimientos -confiabilidad y precisión en las otras dimensiones- y, si los cumple, se devuelve esa instantánea.
    - Si se encuentra la instantánea anterior pero no se satisfacen los requerimientos de las dimensiones no temporales se informa un error de falta de calidad.
    - Finalmente, si no existe una instantánea descriptiva del pasado para la entidad se notifica un error de inexistencia.
      - Otro caso posible, si no existe una instantánea para el pasado y en el requerimiento se solicita que se contemple la posibilidad de especular, es que se cree un recorrido para la entidad en cuestión para que determine dónde podría haber estado la entidad en el momento solicitado por el requerimiento.

- Información del presente
  - Se comienza buscando una instantánea descriptiva del presente en el escenario o una instantánea atemporal en el mapa.
    - Si se halla, se verifica si cumple con los demás requerimientos -confiabilidad y precisión en las otras dimensiones- y, si los cumple, se devuelve esa instantánea. Este caso puede ser muy frecuente cuando más de un recorrido comparten una misma entidad.
    - No obstante, si no cumple con los requerimientos de las otras dimensiones, el SIET realiza una solicitud al SP para mejorar la calidad de los datos de la entidad. En este punto el SIET podría combinar varios requerimientos de distintas fuentes para generar la solicitud al SP de modo incorporar en la solicitud la importancia de

la fuente, la tolerancia temporal y el índice de ejecución de las fuentes. En caso que el SP pueda responder con la calidad demandada antes del vencimiento de la solicitud se registra en el modelo esa información y se responde el/los requerimientos satisfechos. En cambio, si obtiene algo pero de menor calidad se informa falta de calidad y, por último, si no se obtiene nada se informa un error de *time out* a la fuente.

- También puede darse el caso que en el requerimiento se especifique que si no es posible obtener la información de la entidad a partir del sensado entonces se responda con una estimación de la misma. Así, en esa situación, el SIET creará un recorrido llamado igual que la entidad a partir de una evolución de tipo descriptivo cuyo nombre sea igual al nombre del prototipo de la entidad. Una vez creado aquel recorrido entonces se encargará de generar las instantáneas descriptivas estimadas para la entidad mientras tanto no sea posible sensarla. El SIET dejará de utilizar este recorrido cuando retome la información desde el SP ordenando entonces que el recorrido termine ya que no es más necesario.
- Finalmente, también el requerimiento puede incorporar que se considere especular la información de la entidad. En este caso el SIET crea un recorrido especulativo sobre la entidad del requerimiento. El requerimiento también puede especificar la cantidad de instantáneas especulativas que quiere sobre la entidad ya que una evolución especulativa puede, en principio, plantear varias instantáneas en posiciones distintas para una misma entidad en un mismo momento.

- Información del futuro
  - Se comienza buscando en el escenario una instantánea descriptiva del futuro para la entidad que cumpla con el requerimiento de la dimensión temporal. Si se halla, se verifica si cumple con los demás requerimientos -confiabilidad y precisión en las otras dimensiones- y, si los cumple, se devuelve esa instantánea.
  - En caso contrario, el SIET creará un recorrido llamado igual que la entidad a partir de una evolución de tipo descriptivo cuyo nombre sea igual al nombre del prototipo de la entidad. Una vez creado aquel recorrido entonces se encargará de generar las instantáneas descriptivas predictivas para la entidad. El SIET utilizará este recorrido hasta cuando el requerimiento sea desechado por la fuente. En caso que no se pueda cumplir con la calidad demandada o no se pueda cumplir con los requisitos de tolerancia temporal se informará o bien con un error de falta de calidad o con uno de *time out*.
  - El requerimiento también puede especificar que se provea información especulativa si no se puede constituir el recorrido descriptivo predictivo. En cuanto existan en el modelo instantáneas descriptivas que respondan los requerimientos especulativos implica que el recorrido especulativo ya no tiene razón de ser y el SIET lo termina.

Seleccionar las instantáneas procedimentales a ejecutar

El SIET registra e integra lo producido por distintas fuentes y si bien sería conveniente que todas las instantáneas fueran materializadas puede darse el caso que algunas instantáneas no puedan materializarse. Más aún, ciertas instantáneas procedimentales podrían no ser ejecutadas por inconvenientes con los recursos. Los factores que afectan a la ejecución son:

- Incapacidad del SA de transformar las instantáneas declarativas a instantáneas procedimentales.
- La competencia directa entre dos o más fuentes por una entidad propia (inconsistencia).
- Escasez de recursos de procesamiento en general como memoria o procesador.

El primer punto es ajeno al alcance del SIET el cual se limita a recibir un mensaje desde el SA que le informa que no se pudo realizar la transformación de instantáneas y simplemente lo retransmitirá a la fuente de la instantánea para que aquella defina cómo continuar.
En cambio, el segundo y el tercer factor sí afectan a la ejecución y además comparten la misma forma de mitigación. En la sección Materialización y ejecución -pag. 55- se trata con detalle el asunto presentándose aquí sólo una introducción.
La arquitectura ofrece seis elementos en los cuales los algoritmos que deban resolver estos conflictos se puedan basar. Los mismos son:

1. importancia de la fuente
2. hitos de interrupción
3. *processtime*
4. índice de ejecución
5. *time*
6. espacio/tiempo/energía

Lo primero por lo cual debe velar el algoritmo es que las instantáneas de fuentes con una alta importancia no vayan a quedar canceladas. De la mano con la importancia está el *processtime*. Así, si un instantánea tiene una alta importancia y una gran tolerancia temporal entonces no es necesario ejecutarla muy rápido si existen instantáneas de menor importancia que están próximas a vencerse que podrían salvarse si se ejecutan pronto. Por lo tanto, las tolerancias temporales de las instantáneas de alta importancia señalan el límite de cuánto se pueden compartir los recursos genéricos de procesamiento. En medio de la importancia y la tolerancia están los hitos de interrupción que se relaciona con el balance entre planificación y reactividad de acuerdo con las importancias y las tolerancias temporales de las fuentes involucradas.
Una vez que los tres primeros elementos fueron considerados entonces se analiza el índice de ejecución de las fuentes. El sentido es atender primero a las instantáneas de las fuentes con más bajo índice de ejecución con el propósito de tratar de minimizar la inanición[5] de las fuentes de baja importancia. Luego, entre aquellas instantáneas que queden a las puertas de la ejecución, se seleccionan aquellas que su dimensión temporal esté más cercana al tiempo real del robot.
Finalmente se podría realizar una selección de acuerdo a la relevancia que en esas circunstancias la aplicación del robot le asigne al espacio recorrido, el tiempo demandado y la energía a consumir.
Además de lo anterior, ya que la selección implica uso de recursos compartidos es probable que se presenten situaciones de deadlock. La estrategia seguida es prevenir los deadlock buscando ciclos en el grafo de asignación. De modo que cuando se descubre un ciclo una de las instantáneas participantes del mismo se le pospone la materialización.
Es claro que, por más eficaz y eficiente que pudiera ser el algoritmo que controla la selección de instantáneas, será posible que algunas provenientes de las fuentes de menor importancia puedan vencer sus tolerancias temporales. En ese caso se avisa a la fuente de ese hecho ya que la fuente no podrán, de observar sólo la instantánea

[5] Por inanición se refiere al hecho que una instantánea sea constantemente relegada para la ejecución hasta que finalmente se venza con lo cual la fuente no progresa nunca ya que no tien suficiente importancia.

descriptiva, deducir que no se logró ejecutar su prescriptiva por falta de importancia. Así la fuente podrá o bien ampliar su tolerancia temporal para compensar una baja importancia o enviar un mensaje de error a la fuente superior de mayor importancia.

Ofrecer una memoria espacio temporal de entidades

El modelo del SIET en principio registra la existencia y posición de entidades que son caracterizadas por sus prototipos y un identificador de entidad. Sin embargo, algunos prototipos pueden tener características variables -ver en el sistema de modelización de objeto (SMO)- lo que determina que el valor que tomen las características variables se deba obtener durante el funcionamiento del robot -casi siempre por medio del sensado-. Por lo tanto, como los valores pueden cambiar dinámicamente y están asociados a entidades, entonces la arquitectura establece que se registren con las instantáneas de las entidades correspondientes. Por ejemplo, un semáforo de un único foco, puede mostrarse en rojo, amarillo o verde pero en el prototipo del mismo no se puede establecer ningún color en base a un criterio firme así que se define como una característica variable con valor indefinido. Luego en el funcionamiento del robot se determinará cuál de los tres valores se presenta, por medio del sensado, y se registrará en las instantáneas que correspondan.
Así, dependiendo del tipo de entidad y del uso que se haga de esta, los valores de las características variables estarán registrados espacio-temporalmente por el SIET y este podrá ofrecerlos a los demás sistemas mientras se encuentren en el modelo.
Además por medio de las entidades virtuales los demás sistemas pueden "*recordar lugares*" de interés para los procesos que lleven a cabo. Finalmente, las entidades abstractas, sirven para mantener un estado compartido entre distintos sistemas y/o fuentes sobre el cual además pueden llevarse a cabo consultas temporales. Esto es, se puede preguntar el valor que mantiene la entidad abstracta a distintos momento de tiempo.

Presentar su información de acuerdo a un marco de referencia solicitado

Al inicio de esta sección se mostró que el SIET registra las fuentes que usan las entidades para poder mantener acotado el tamaño del escenario. Aquí se verá que las fuentes también se ven beneficiadas de este conocimiento de que dispone el SIET. Básicamente cuando una fuente se declara en el SIET le informa a este una entidad de referencia espacial. Esto significa que el SIET cuando le notifique las posiciones de las instantáneas lo hará con respecto a aquella entidad de referencia y de modo recíproco cuando la fuente le informe la posición de las instantáneas que produzca también lo hará bajo aquel marco de referencia.
El propósito de que cada fuente elija una entidad como referencia es simplificar sus algoritmos internos al poder expresar las relaciones entre las entidades del modo más natural al problema que resuelve la fuente. Por ejemplo, si una entidad debe girar alrededor de otra siguiendo una órbita lo natural es usar la entidad del centro como entidad de referencia y no, una tercera entidad, que hará que las expresiones que determinan la posición de la instantánea que orbita sean muy complejas.

Resolver consultas sobre relaciones espacio-temporales entre entidades

El SIET contiene las relaciones espacio temporales del modelo del mundo del robot y, cómo se desprende de lo visto, son las fuentes las usuarias de las mismas consistiendo ese uso en evaluar situaciones espacio-temporales para determinar cómo generar sus instantáneas.
Por lo tanto, como sería un despropósito repetir muchas veces los mismos algoritmos de evaluación en las distintas fuentes es que la arquitectura propone establecer un motor de algebra espacio-temporal que resuelva las consultas de las fuentes como un servicio.

Así las fuentes plantean sus necesidades sin tener que preocuparse en cómo se resolverán las mismas. Además, separar este motor de las fuentes permite que el mismo siga un desarrollo de software paralelo y que las mejoras se vean inmediatamente aprovechadas por todas las fuentes.
Por el lado de la implementación, el hecho que el motor esté en el mismo sistema en donde está el modelo posibilita que el primero tenga accesos al segundo de un modo más eficiente que el que se lograría si los accesos debieran hacerse a través de la interfaz pública.

Determinar las entidades próximas del robot

Las entidades que se mantienen en el escenario son aquellas que están siendo utilizadas por alguna fuente. Sin embargo, las entidades no propias que están más cerca de la entidad *Self* son más importantes que aquellas que están más distantes ya que las primeras pueden estar involucradas en la acción del robot.
Es por eso que el SIET brinda un servicio de notificación de novedades del entorno cercano del robot. El mismo consiste en notificar ya sea cuando una entidad no propia se aproxima más allá de una distancia como cuando la entidad se aparta más lejos que el límite del contexto inmediato. Esto es especialmente útil para el sistema de actuación, tal como se explica en la respectiva sección, para aplicar los costosos cálculos llevados a cabo por su motor de dinámica sólo a las entidades que tienen sentido práctico hacerlo.

Establecer un umbral de planificación

Las fuentes, según cómo estén realizadas, pueden considerar llevar una pequeña planificación de instantáneas para un muy breve futuro -ver en “Sistema de motorización del lenguaje (SML)”-. No obstante, se puede presentar la circunstancia que las instantáneas planificadas finalmente cuando deban materializarse diverjan mucho de las efectivamente materializadas. Es por eso que si una fuente planifica instantáneas con un “gran” horizonte en una situación inestable puede llegar a ser un verdadero desperdicio de capacidad de procesamiento.
Para subsanar este inconveniente el SIET en base a la actualización de las instantáneas calcula para cada fuente cuantas instantáneas puede planificar en forma que no diverja mucho lo planificado con lo materializado y se lo pasa a la fuente que entonces planifica sólo hasta esa cantidad. Cada tanto tiempo, para evitar converger a la reactividad pura -i.e. con ninguna instantánea planificada-, el SIET incrementa el umbral para verificar si ya se ha abandonado la situación inestable y así con ese incremento del umbral proponerle a la fuente que vuelva a realizar planificaciones más extensas.

Procesos del SIET

En esta sección se describen los procesos que son realizados por el SIET para llevar a cabo su funcionamiento.

Materialización y ejecución

La materialización es un proceso en que intervienen varios sistemas y en donde el SIET lleva el papel de intermediar entre estos. La intervención de los sistemas es por medio de fuentes coadyuvantes y el SIET a través de su módulo de dispatcher resolverá los conflictos que pudieran ocurrir. A continuación se describen las fuentes coadyuvantes y el dispatcher.

Fuente coadyuvante

Las fuentes coadyuvantes[6] son las fuentes que operan en los sistemas percepción y actuación con el fin de resolver un requerimiento descriptivo o materializar una instantánea prescriptiva declarativa respectivamente.

[6] *Derecho*. Persona que interviene en un proceso sosteniendo la pretensión de una de las partes.

Independiente si es del SP o del SA todas estas fuentes coadyuvantes comparten un mismo comportamiento general diferenciándose en las del SP que seleccionan sensores y generan como producto instantáneas descriptivas y las del SA que seleccionan actuadores y que su producto son instantáneas prescriptivas procedimentales. A continuación se definen unos términos que se usarán en el resto de la sección.

> **Fuente propósito:** es la fuente que necesita de una fuente coadyuvante para cumplir con su fin. Las fuentes propósito pueden estar tanto en el SML como en el SP y en el SA.
> **Entidades objetivo**: son tanto las entidades que se necesitan que sean sensadas como aquellas que se necesitan que sean movidas; así, respectivamente son llamadas ***entidad objetivo descriptiva*** y ***entidad objetivo prescriptiva***.

Así una fuente propósito del SP o del SA puede necesitar una fuente coadyuvante del SP cuando necesitan conocer la posición de un sensor y en el caso del SA también cuando necesita informarse de la posición de un actuador. En la oración anterior se utilizó el verbo pueden ya que sobre las entidades propias existen sensores de propiocepción -encoders por ejemplo, ver sistema de gestión de hardware y de actuación- que no necesitan ser sensados por el SP. Asimismo una fuente coadyuvante del SP puede ser también la fuente propósito de una fuente coadyuvante del SA cuando la primera necesita mover un sensor. Por ejemplo, si se necesita saber la posición de un brazo que en el extremo tiene una sonda se puede utilizar los sensores de propiocepción de las coyunturas del brazo pero también se puede obtener a través de otra fuente coadyuvante por medio de un sensor cámara que necesitará ser desplazado para que el SP determine dónde está el brazo en cuestión.
Las fuentes coadyuvantes mantienen la integridad de los propósitos originales que tienen que resolver incorporando en la materialización de las instantáneas la importancia y la tolerancia temporal de las entidades objetivo.
Con respecto a la tolerancia temporal la misma es descompuesta entre todas las instantáneas procedimentales que se necesiten generar para implementar lo solicitado por la instantánea declarativa de la entidad objetivo. Todas las instantáneas generadas por las fuentes coadyuvantes portan además como metadato el identificador de la fuente propósito del SML original para que cuando se resuelvan los posibles conflictos de asignación de recursos no se divida la ejecución haciéndola inviable. En las secciones del SA y del SP se detalla el comportamiento de las respectivas fuentes coadyuvantes.
Por último, cada instantánea procedimental contiene el identificador de la instantánea declarativa de modo que el SIET pueda aplicar al tren de instantáneas procedimentales las restricciones de ejecución de la declarativa que materializan.

Dispatcher

La función del dispatcher es analizar las instantáneas prescriptivas procedimentales y su relación con las configuraciones de hardware con el propósito de velar que no haya conflictos de inconsistencia -por ejemplo, una fuente pide mover una entidad en sentido y otra fuente en el sentido contrario- como así también problemas con la asignación de recursos con la presencia de deadlocks. Ambos tipos de conflictos son resueltos por el dispatcher cuando selecciona qué instantáneas procedimentales serán integradas y cuáles deben ser rematerializadas por sus fuentes coadyuvantes.

Identificación de conflictos de inconsistencias

La identificación de los conflictos de inconsistencia trabaja sobre las instantáneas prescriptivas propias ya que en estas está lo que el robot hará en el futuro cercano. Así, no se tienen en cuenta las entidades no propias dado que estas importan para el movimiento del robot si interactúan con este y esa posible interacción ya queda

reflejada en las instantáneas prescriptivas propias. Con respecto a las descriptivas propias tampoco hay que considerarlas en los conflictos de inconsistencias porque las prescriptivas operan siempre tras una descriptiva propia. Por último, cabe aclarar que las instantáneas especulativas no existen para las entidades propias.
Una inconsistencia se produce cuando diferentes fuentes establecen para un mismo momento y entidad instantáneas prescriptivas con destinos espaciales distintos. Por ejemplo, para una entidad que represente una cámara una fuente podría necesitar apuntarla hacia el horizonte para validar el rumbo correcto del robot pero en cambio otra fuente podría necesitar que apunte adelante en el piso para verificar que no haya obstáculos.
Es claro que un conflicto sólo surge si existe más de una fuente con pretensiones sobre una misma entidad propia por eso ese es el primer criterio a verificar: si sólo hay una fuente para la entidad propia entonces no hay inconsistencias.
Para el caso que sí haya más de una fuente involucrada, entonces cualquiera de los siguientes puntos a analizar que sea verdadero implica la presencia de un conflicto a resolver:

- que las instantáneas procedimentales intervinientes tengan configuraciones de hardware distintas. Esto implica que las fuentes tienen diferentes necesidades; para una fuente puede significar moverse con precisión aunque lento y para la otra fuente moverse rápido sin importar la precisión.
- que la distancia entre las instantáneas sea mayor a un porcentaje del tamaño de la entidad -el porcentaje en cuestión es un parámetro del sistema- y que la diferencia entre la dimensión temporal entre estas sea menor a una constante que es un parámetro del sistema. Esto es, que cada fuente pretenda que esté la entidad en dos lugares distintos al mismo tiempo.
- que la diferencia del módulo de la velocidad lineal de las entidades sea mayor que un porcentaje de la menor velocidad de las involucradas. Este porcentaje también es un parámetro del sistema. Para obtener la velocidad se consideran dos instantáneas sucesivas de cada fuente. Esto significa que si las fuentes tienen tienen expectativas muy diferentes acerca de la velocidad de movimiento de la entidad entonces hay un conflicto.
- que la diferencia de la velocidad angular para cada una de las tres dimensiones de la actitud sea mayor que un porcentaje de la menor velocidad angular de las involucradas en la dimensión. El porcentaje es un parámetro del sistema. Esto significa que si cada fuente pretende que gire la entidad a diferentes velocidades entonces hay un conflicto.

Por otro lado, si los cuatro puntos son falsos, entonces no existe conflicto y se unifican las instantáneas marcando a la instantánea de menor importancia como dependiente. Eso implica que la instantánea dependiente libera las configuraciones de hardware que necesitaba.

Resolución de conflictos

En caso que haya un conflicto la resolución del mismo se organiza según las siguientes fases y criterios de prioridad:

1. Fase 1
   - importancia de la fuente,
   - hitos de interrupción,
   - tolerancia temporal -*processtime*-,
2. Fase 2
   - índice de ejecución
3. Fase 3
   - dimensión temporal -*time*-,
4. Fase 4

- interés en cuidar espacio o tiempo o energía[7].

Definición de plan

En este contexto existe un plan si se presenta una secuencia de instantáneas desde una misma fuente que finalizan con un hito de interrupción. Si no existe tal hito entonces es una fuente reactiva probablemente con muy pocas instantáneas y no se considera sea un plan. Los planes tienen la misma importancia que su fuente.

Fase 1

En la primera fase se utilizan los criterios de importancia, los hitos de interrupción y la tolerancia temporal para seleccionar los planes a integrar en la ejecución. El criterio para aceptar que un plan menos importante a otros más importes se ejecute es que se pueda ejecutar sin comprometer la ejecución de otro plan más importante. Esto es, si existe tiempo suficiente para ejecutar primero el plan menos importante y luego volver el actuador a la posición inicial de dónde comenzó el plan menos importante antes que se venza la primera instantánea del plan más importante. La solución trivial para la anterior evaluación es considerar que el plan es simplemente reversible de modo que se suponga que tomará al actuador el mismo tiempo volver al origen que lo que le tomó ejecutar el plan. Sin embargo, en muchos casos, eso no es cierto; por ejemplo, levantar un objeto es distinto a bajarlo. A pensar de lo sencillo de la solución trivial, tal enfoque puede funcionar -sobre todo como prueba para la idea- ya que lo que se busca es determinar una cota temporal y no efectivamente volver a la posición de partida inicial.

Lo subyacente es que si la fuente objetivo más importante no es la que efectivamente logre ejecutar sus instantáneas -ya que el dispatcher le cedió su lugar a una fuente menos importante- entonces la fuente más importante se notificará de esa situación por medio de las instantáneas descriptivas cuando no coincida lo descriptivo recién actualizado con lo que prescribió con anterioridad. Así, como en cada momento, se evalúa toda la situación cuando se termine el plan de menor importancia el plan corriente de la fuente de mayor importancia probablemente estará armado a partir del final del plan de menor importancia. De ahí es que probablemente no haga falta "volver" al punto original sino que directamente continúe con el plan que corresponda según el criterio establecido.

Fase 2

Si todos los planes pueden ser cubiertos en la fase anterior, entonces los demás criterios son considerados para decidir el plan a llevar a cabo. El siguiente punto a tener en cuenta es tratar de mantener el índice de ejecución lo más cercano a cero con el fin de mitigar la inanición de las fuentes. Por eso, si hay una fuerte dispersión de valores de índices de ejecución entonces, se seleccionará para integrar primero la fuente con peor índice de ejecución. Sin embargo, si todos los valores del índice de ejecución están cercanos al cero entonces se sigue con el próximo criterio.

Fase 3

El ante último criterio es por la dimensión temporal dónde se seleccionará a aquella fuente cuya primera instantánea esté más próxima al momento presente del tiempo real y teniendo en cuenta cierta cota que es un parámetro del sistema. Si ninguna fuente es seleccionada por este criterio se continúa con el último criterio.

Fase 4

El último criterio trabaja sobre el interés que le importa optimizar al robot. Dependerá que se puedan calcular para cada permutación en el orden de ejecución de los planes conflictivos los factores que importen. No obstante, si bien es exponencial en la cantidad de micro planes, no se espera que haya muchos micro planes en conflicto. El más sencillo método de selección es entonces elegir al azar cual se ejecutará primero

[7] En realidad es cualquier función que se pueda calcular sobre los planes. El camino más interesante en una exploración de un terreno podría ser justamente el que no optimice ni espacio, ni tiempo, ni energía.

si bien se pueden construir muchas heurísticas para que no se desperdicie toda la posible optimización en el punto de interés.
Así el hecho interesante sobre el funcionamiento del dispatcher es que cuando la tolerancia temporal es poca se concentra en las actividades más importantes y que cuando la tolerancia temporal es holgada permite tomarse un tiempo para optimizar su comportamiento.
Una consecuencia de la definición de plan es que las fuentes reactivas al no tener hitos de interrupción su ejecución depende exclusivamente de su prioridad y por no tener puntos de corte explícitos no deben ejecutarse cuando tengan menos importancia que otros planes de mayor importancia ya que no existe la garantía de que les ceda el paso. Por eso, las fuentes reactivas sólo pueden ejecutarse cuando logran el primer lugar de importancia.

Algoritmo del dispatcher

A continuación se describe la idea del algoritmo del dispatcher:

- El ciclo principal encola los planes que vayan llegando y dispara, cada tanto tiempo, el bloque que construye y ejecuta un cronograma -i.e. un conjunto de planes cada uno con un momento definido en el cual se ejecutará-. En ese momento pueden seguir arribando planes, pero serán encolados y no formarán parte de ese cronograma en construcción.
  - Se ordenan los planes a considerar por importancia.
  - Se inicializa una lista que sera la estructura para el cronograma en construcción.
  - Los planes se recorren según el orden de prioridad y para cada plan se lleva a cabo:
    - ¿Tiene el plan corriente un hito de interrupción?
      - Sí tiene hito de interrupción:
        - ¿Es posible ejecutar el plan corriente sin comprometer el comienzo de la ejecución de todos los planes agregados en el cronograma?
          - Sí es posible:
            - se agrega el plan corriente al final del cronograma.
          - No es posible:
            - se marca como terminada la construcción del cronograma.
      - No, no tiene hito de interrupción -i.e. es reactivo-:
        - ¿Está el cronograma aún vacío?
          - Sí está vacío:
            - se agrega el plan corriente al cronograma.
          - No, no está vacío:
            - se marca como terminada la construcción del cronograma.
    - ¿Se terminó la construcción del cronograma?

- Sí, se terminó:
  - ¿Se procesaron todos los planes?
    - No, no se procesaron todos los planes:
      - Se envía a la fuente del plan corriente y a todos las fuentes de los planes aún sin evaluar un mensaje de error informando que no tienen suficiente importancia para integrarse el cronograma de ejecución.
    - Sí, se procesaron todos:
      - En función del tiempo disponible para comenzar a ejecutar los planes y la cantidad de planes en el cronograma se reordena la lista del cronograma armada para así aplicar los demás criterios.
- No, no se terminó:
  - Avanzar la iteración para procesar el siguiente plan en importancia.

- Recorrer la lista del cronograma construido desde el plan de menor importancia al de mayor:
  - Se envían al SA las instantáneas procedimentales del plan.

Resolución de conflictos de asignación de recursos

Además del conflicto tratado en el punto anterior, que es cuando más de una fuente reclaman una misma entidad o configuración de hardware, también se puede presentar un problema de deadlock. En la formulación clásica de este problema las fuentes se inhabilitan entre sí en la ejecución de sus planes porque esperan un recurso que tiene tomado otra fuente que está en deadlock y, que a su vez, esta espera un recurso que también está tomado por una fuente perteneciente al deadlock. Esto conlleva a una espera circular sin que ninguna fuente del deadlock progrese. Sin embargo, puesto que cada nodo [orden en cola] está asociado a una instantánea prescriptiva y la misma tiene un vencimiento por más que no se atienda el problema de deadlock el sistema podría progresar -si bien de forma lejana a la óptimo- ya que los vencimientos liberan los pedidos realizados.

Así, por una cuestión de eficiencia, se tratan estos deadlock temporales. La propuesta utilizada aquí se sirve de una técnica de detección de deadlock, sin embargo, como se verá un poco más adelante, no sólo detecta los deadlocks presentes sino que también en la práctica podrá prevenirlos y disolverlos antes de que ocurran. El dispatcher verifica la presencia de deadlock y ante la presencia de uno libera -como mínimo- los recursos de la fuente de menor importancia perteneciente al deadlock de modo que todas las demás fuentes puedan continuar.

La verificación consiste en detectar un ciclo en el grafo formado a partir del grafo del SIET de la siguiente manera:

- todos los nodos de requerimientos descriptivos y de las instantáneas prescriptivas de una fuente coadyuvante son representados por un único nodo que representa al proceso de la fuente coadyuvante.
- todos los arcos dirigidos de asignación, de espera de recurso y de pedido de recurso son representados por un mismo tipo de arco pero manteniendo el sentido original.

- los nodos [orden en cola] de las colas de las configuraciones de hardware se mantienen sin modificar representando los recursos.

Como ya se adelantó, la detección de ciclos en el grafo es un método de prevención ya que como actúa sobre las colas de espera antes de que los recursos sean asignados da posibilidad al dispatcher a cancelar los pedidos para evitar los posibles deadlocks.

Instantáneas prescriptivas procedimentales integradas

Finalmente las instantáneas prescriptivas procedimentales integradas serán aquellas que fueron seleccionadas por la verificación de consistencia, no estén en un deadlock y que además tengan un nodo en una cola de espera que esté en la cabecera que no esté esperando a otro nodo de otra cola.

Cuando se cumple con la anterior condición el dispatcher cambia la relación de pedido de recurso por la de asignación y se notifica al SA sobre las instantáneas integradas y no integradas.

Reconocimiento activo

El reconocimiento activo se pone en marcha cuando una entidad del mapa entra dentro del escenario por el "desplazamiento" del mismo sobre el mapa y la entidad en cuestión no tiene ninguna relación de [necesidad]. De dejarse así el sistema de percepción no la sensará y es por eso que el SIET genera una fuente coadyuvante con una importancia más baja que la más baja proveniente del sistema de motorización del lenguaje. Si durante el reconocimiento activo surge una fuente con interés por la entidad entonces el SIET termina a la fuente coadyuvante que inicio. Finalmente, cuando dicha entidad abandone el escenario el SIET terminará con la fuente del sensado activo.

Determinar el contexto de acción

El motor de dinámica es un componente del sistema de actuación que en base a las leyes físicas de la dinámica calcula los efectos sobre las entidades. Puesto que esto es computacionalmente un trabajo muy intensivo y sólo tiene sentido cuando es posible un contacto del robot con la entidades en cuestión es que el SIET determina en todo momento las entidades involucradas. La implementación más sencilla se sirve de calcular la distancia a las entidades con respecto a *Self*. Así cuando está por debajo de cierto valor de parámetro que determine el tamaño del contexto de acción le avisará al SA que la entidad en juego está dentro del contexto de acción y recíprocamente cuando la distancia de una entidad del contexto esté más allá de este valor de este parámetro el SIET le avisará que al SA que no realice más los cálculos dinámicos para la misma.

Calcular umbral de planificación

El cálculo del umbral de planificación se basa en las actualizaciones que hacen las fuentes sobres sus instantáneas. Se calcula para cada fuente cuántas instantáneas puede planificar de forma que no diverjan mucho lo planificado con lo ejecutado y se lo pasa a la fuente que entonces planifica sólo hasta esa cantidad. Una forma de evaluar la divergencia es almacenar en las instantáneas el primer valor planificado y luego, cuando llega la ejecución o la eliminación, se calcula la distancia entre la posición original y la final. Se la compara con el tamaño de la entidad en cuestión y si el porcentaje es mayor que un parámetro se marca que la planificación no pudo llevarse a cabo correctamente. Cada tanto tiempo, para evitar converger a la reactividad pura se incrementa el valor devuelto para explorar el valor máximo que es posible de planificar en cada momento.

Motor de álgebra espacio temporal

Las funciones del motor de álgebra espacio temporal son encapsular los algoritmos que acceden al mapa y al escenario y brindar una interfaz conceptual de los mismos. La idea de usar este motor es que las fuentes puedan plantear consultas espacio

temporales de forma declarativa con un lenguaje de operadores espaciales. Así se podría directamente solicitar mediciones espaciales como área, distancia, o perímetro. Puesto que el SIET maneja un pequeño ámbito temporal este motor permitirá consultas sobre cambios de medidas espaciales recién hechos o que estén planeados a realizar.

Sistema de modelización de objetos (SMO)

Objetivo

Centralizar la definición de los prototipos de las entidades y los servicios de información sobre los mismos con el fin que los demás sistemas puedan procesar las entidades sin la necesidad de disponer de cómo estas están constituidas.

Introducción al SMO

Básicamente el SMO es un sistema de base de datos de objetos 3D con un motor de renderizado [2][4][10]. Como base de datos su función principal es encontrar los prototipos que cumplan con criterios de búsqueda que le planteen los otros sistemas sobre las definiciones de los prototipos y como motor de renderizado tiene dos funciones. La primera es generar patrones comparables con los resultados de los sensores. Esto es que los patrones generados en la renderización sean una simulación de lo producido por un sensor real. Así, vez de hacer un análisis de los datos sensados, se realiza una síntesis de lo esperado a partir de la definición del prototipo y se compara con lo sensado. La segunda función es construir una máscara "transparente" en el patrón de comparación generado para que en la zona de la máscara se muestree lo sensado en vez de comparar el valor sensado con lo esperado. Esta es la forma en que se puede obtener un valor sensado.
Puesto que el robot en sí es un prototipo es también en el SMO en dónde se definen las capacidades de sensado y actuación del robot de forma que los algoritmos del sistema de percepción y actuación puedan determinar cómo sensar o actuar sobre las entidades. En lo que resta de la sección se definirán algunos conceptos y se tratarán los temas introducidos.

Definición de conceptos

Característica

Las características son esencialmente tipos de datos elementales definidos por el diseñador del sistema robótico que son utilizadas para describir a los prototipos y/o sus partes constituyentes. Las características se califican en abstractas y físicas -abstract y physical en el lenguaje del framework-. Las abstractas tienen como fin representar estados de las entidades. En general, las aplicaciones las usan como memoria de trabajo y no difieren de un tipo de datos tradicional. El propósito de las físicas en cambio es brindar un estándar para los valores devueltos por los sensores y de ese modo poder describir los prototipos de forma independiente al sensor utilizado. Algunos ejemplos de características físicas pueden ser: reflectividad IR, reflectividad en luz visible, absorción del sonido de superficie, masa, fricción, color, temperatura, etc.

Prototipo

Son definiciones que describen los objetos[8] que podrían existir en el ámbito de actuación de destino para el robot.
Existen tres tipos de prototipos: los concretos, los virtuales y los abstractos. Los primeros son para denotar objetos que pueden ser sensados y/o que se puede actuar sobre estos. Los virtuales son para representar entidades como puntos o zonas en el espacio y/o momentos, útiles para referencias espacio temporales pero sin existencia

[8] Aquí el término objeto hace alusión directa a los objetos del mundo y no a la entidad de programación denominada objeto. Aunque como se verá más adelante los objetos abstractos son prácticamente registros o estructuras en el sentido de programación.

física. Finalmente los abstractos están para representar los estados globales del robot en lo referente a la aplicación, por ejemplo en una aplicación de fútbol de robots podría ser el marcador del juego. De estas diferencias anteriores las entidades derivadas de los prototipos concretos y virtuales dispondrán de las siete dimensiones -rumbo, altura, distancia, cabeceo, alabeo, guiñada y tiempo- utilizadas para posicionar completamente un objeto en el espacio tridimensional temporal pero en cambio las entidades derivadas de los prototipos abstractos sólo contendrán a la dimensión tiempo.

Prototipos no abstractos

Los prototipos no abstractos -i.e. concretos y virtuales- son construidos incrementalmente a partir de elementos y/o con otros prototipos definidos con anterioridad. Los elementos son los componentes más primitivos y los mismos se especifican por medio de una geometría NURBS[9] construida externamente. Así, de los prototipos ya definidos y de los elementos es posible construir nuevos prototipos.

Con miras a las futuras ampliaciones de la arquitectura -cuando el robot pueda modelar sus propios prototipos- se define que los objetos deben ser definidos desde el lenguaje y por el momento sólo habrá una instrucción primitiva que cree los elementos a partir de un archivo de geometría NURBS.

Todo el sistema de definición de objetos no abstractos se basa en que la definición de un objeto se lleva a cabo con la agregación de objetos/elementos partes posicionándolos en un espacio centrado en el objeto que se está definiendo. Así cada definición de prototipo es un sistema de coordenadas. Cuando a su vez el objeto definido sea utilizado como parte de otra definición entonces su sistema coordenado servirá para la orientación de la parte en el sistema coordenado al cual se agrega.

Prototipos concretos

El prototipo concreto es un prototipo no abstracto que tiene al menos una característica física. Las características pueden estar asociadas a los elementos que directa o indirectamente construyen el prototipo o también pueden ser propiedades globales del prototipo.

Prototipos virtuales

Un prototipo virtual define un volumen, un plano, un punto y/o un momento del espacio tiempo. De esta forma se permite al robot integrar los objetivos de desplazamiento como entidades a su modelo. Por ejemplo, si el robot se tiene que desplazar y después volver al punto de partida puede crear una entidad virtual punto para la posición actual y después para volver sólo tiene que acercarse a dicha entidad virtual. Formalmente, un prototipo es virtual, si es no abstracto y no tiene ninguna característica física.

Prototipos abstractos

Un prototipo es abstracto si no tiene una representación espacial. Además uno abstracto sólo puede tener características abstractas y son muy similares a los *records* o a las *structs* de los lenguajes como Pascal o C. Se diferencian de los *records* porque contienen la dimensión tiempo implícitamente definida más allá de qué características abstractas utilice con el fin de garantizar que todas las entidades abstractas sean gestionadas temporalmente. Por último, mientras que los prototipos abstractos sólo pueden tener características abstractas los no abstractos pueden tener también características abstractas.

[9] Acrónimo inglés de la expresión *Non Uniform Rational B-Splines*. Son utilizadas por las aplicaciones de diseño ya que permiten especificar cualquier forma con precisión, consumen menos memoria que las representaciones alternativas, tienen una forma de evaluación relativamente eficiente y existen estándares para intercambiar las geometrías NURBS entre los diversos sistemas.

Prototipo self

La naturaleza genérica de la arquitectura hace que el propio robot sobre la cual la arquitectura esté implementada deba ser modelado con un prototipo concreto y su instancia correspondiente. Para eso la definición del prototipo se calificará con la palabra reservada *self* para denotar este carácter especial y, a su vez, en las evoluciones *Self* -con mayúscula- será la instancia que represente al robot y que será creada automáticamente junto con *Main*. La idea de que el robot sea una colección de entidades es lo que permite a los sistemas de la arquitectura operar sobre este de forma uniforme a cómo lo hacen con las demás entidades.

Subentidad

Las subentidades son entidades que están definidas en un estado de "letargo" en los prototipos. Cuando el prototipo que las contiene es utilizado para crear una entidad entonces las mismas estarán disponibles. Sin embargo, a pesar que estén disponibles la identificación de las mismas dependen de la entidad principal debiéndose acceder por navegación a las subentidades. Las consultas sobre prototipos pueden predicar sobre sus subentidades como así también las consultas al motor del álgebra espacio temporal del SIET. Por ejemplo, supóngase que es necesario hacer un prototipo de un carro con cuatro ruedas. En ese caso se realizará por un lado un prototipo rueda para describir las propiedades de las ruedas y por otro lado se construirá el prototipo carro que tendrá cuatro subentidades rueda cada una de estas con un identificador dependiente del identificador de entidad de las futuras instancias del prototipo carro. Así cada vez que se crea un entidad carro también se crearán las cuatro entidades rueda. No obstante la existencia de un identificador dependiente de la entidad padre, las subentidades tienen un atributo "*nombre subentidad*" que las identifica en el contexto del prototipo del cual forman parte. Esto es especialmente útil para poder seleccionar prototipos o relacionarlos aún cuando no haya instancias.

Además, las subentidades contenidas en el prototipo *self* deben tener unos calificadores que carecen los demás prototipos; estos son -en términos utilizados en el lenguaje-: *sensor*, *actuator*, *pasive* y *range*. La función de los mismos es precisamente informar a los sistemas qué estas entidades son especiales - agrupándolas en sensores, actuadores, las que constituyen al armazón del robot y las virtuales que determinan el espacio cubierto por el alcance de los sensores - para que estos funcionen diferenciadamente cuando sea necesario. Estos calificadores no sólo pueden utilizarse en la creación de las instancias dentro de *self* sino que también en la definición de prototipos contenidos dentro de *self*. El default para este calificador en los prototipos distintos de *self* es *foreign* y por ser default no es necesario de expresarlo en el código del lenguaje.

Configuración de hardware

Las configuraciones de hardware son la abstracción que lleva a cabo la arquitectura para denotar los modos de funcionamiento del hardware de los actuadores y sensores que son útiles a la aplicación del robot. Para el SMO todo el hardware del robot se reduce a los drivers de control del mismo que presenta el SGH y una configuración de hardware es precisamente un ajuste de los parámetros de un driver.

El propósito final de las configuraciones de hardware es que el robot pueda experimentar sobre las capacidades de su hardware -o adaptar su uso a medida que se desgasta- con el fin de poder tener más herramientas para afrontar situaciones inesperadas. No obstante, en una primera aproximación, las configuraciones de hardware sólo harán referencia al nombre del driver utilizado y soslayará todos los procesos necesarios para comunicar los parámetros y los valores máximos y mínimos e interacciones entre los dichos valores que los drivers pudieran tener así como toda la capacidad de experimentación en el uso del los drivers. De ahí que cada driver que actualmente se utilice implicará un ajuste de los parámetros que debe realizar el

diseñador del robot, esto es, las configuraciones de hardware serán inicialmente definidas externamente.

Zona de sensado

Las zonas de sensado son un constituyente del modelo del SMO cuya función es integrar los sensores con su capacidad de sensado y el espacio cubierto por ellos. En base a esta información y las necesidades de sensado que surjan los sistemas de la arquitectura seleccionarán a los sensores.
Se compone de los siguientes elementos:

- una subentidad sensor,
- una configuración de hardware del sensor,
- una característica física sensable por el sensor,
- una subentidad virtual, del prototipo del sensor en cuestión, cuyo fin es determinar una región del espacio solidaria[10] a la subentidad sensor que coincide con el alcance del campo sensible del sensor,
- la resolución espacial por cada una de las dimensiones de la subentidad virtual,
- la confiabilidad de que el valor sensado esté espacialmente en dónde lo informa el sensor para cada una de las dimensiones de la subentidad virtual.

Plantilla de comparación y máscara de sensado

Las plantillas de comparación y las máscaras de sensado son el resultado del motor de renderizado. Básicamente son una representación de lo que se espera sensar para determinar si lo sensado se corresponde con lo esperado. Ambas son luego pasadas al sistema de gestión de hardware para que el mismo opere en función de ellas. Las plantillas son exclusivamente para comparar la salida de los sensores, en cambio, las máscaras definen regiones dónde, en vez de comparar, se muestrearán los valores sensados. Esta última es la forma en que las características variables obtienen sus valores.
La constitución de una plantilla es compatible con el dominio de la función sensor -ver SGH-, es decir, un conjunto de tuplas que están conformadas por:

- un escalar para la coordenada de cada dimensión,
- el nombre de una característica,
- un valor válido para la característica anterior.

En la máscara en vez de un conjunto se devuelven dos. El primero es un conjunto de identificadores de variables y el segundo es un conjunto muy similar la salida de la plantilla pero cambiando el valor literal del dominio de la característica por un identificador de variable que pertenezca al primer conjunto.

Descripción general

Como se mencionó el objetivo del SMO es centralizar la modelización de los prototipos de las entidades y los servicios de información sobre los mismos con el fin que los sistemas de percepción (SP), de actuación (SA), de gestión de hardware (SGH) y de motorización del lenguaje (SML) puedan procesar las entidades sin la necesidad de disponer de cómo están estas constituidas.
En la figura 2 se muestra un diagrama del modelo de datos utilizado en el SMO y después se describen las funciones llevadas a cabo por este sistema.

[10] Por solidaria se entiende que la región acompaña el movimiento de su sensor como si fuera un apéndice rígido.

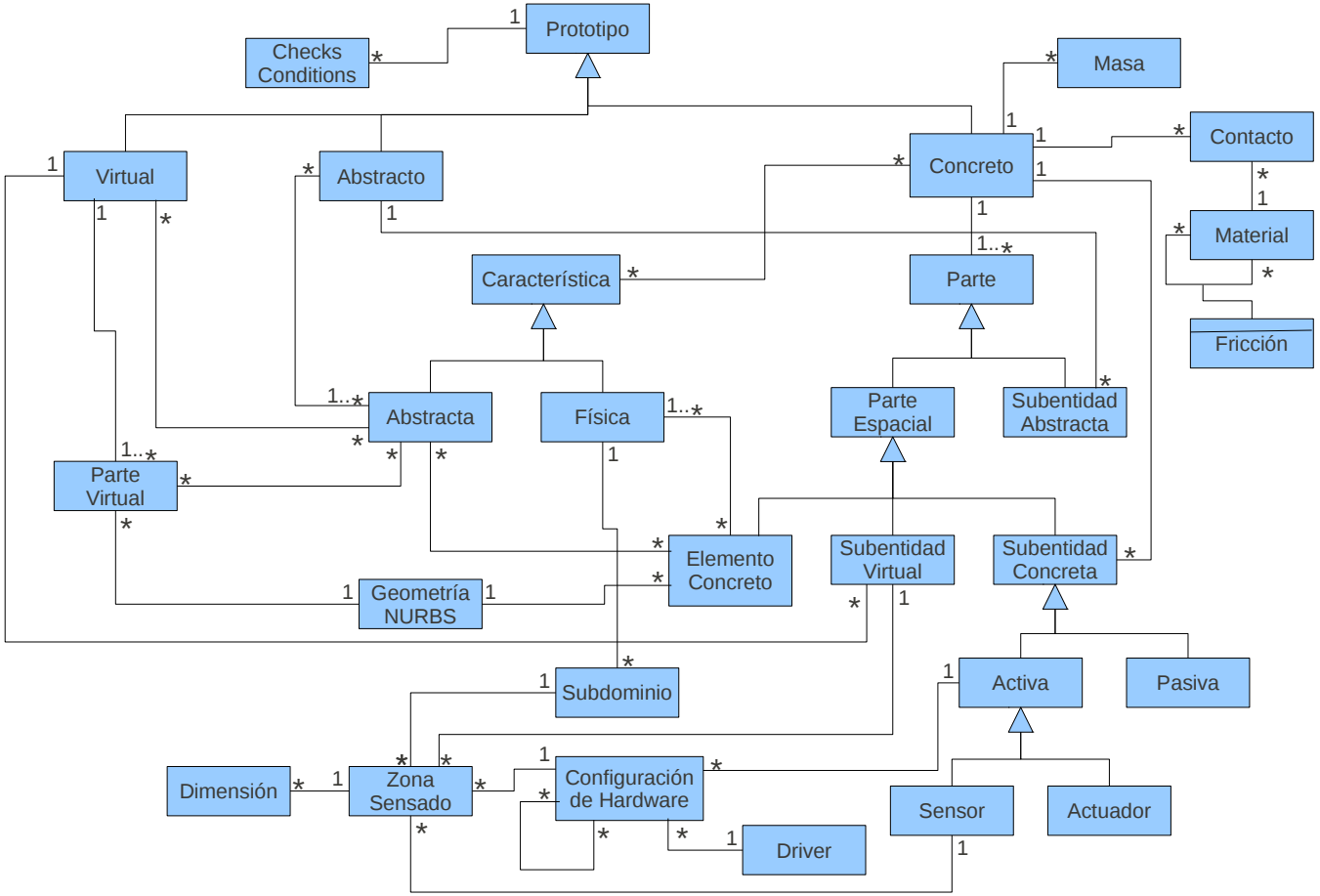


Figura 2 - Diagrama del modelo de datos utilizado en el SMO.[11]

[11]Los valores default para los calificadores *fix*, *constant* y *variable* así como sus valores *overwrited* no están representados en el diagrama.

Registro de prototipos

Los prototipos son identificados por un nombre y son definidos por un bloque de código del lenguaje de la arquitectura. La base de datos del SMO que contiene la definición de los prototipos es persistente a los reinicios del robot por eso el motor del lenguaje del SML cuando encuentra el uso de un prototipo verificará consultando al SMO sobre la existencia del mismo. En caso que el SMO responda que no está registrado entonces el motor del SML buscará en el código del lenguaje el bloque que define el prototipo para ejecutarlo y así crear el prototipo en el SMO. Obviamente si tal bloque de código tampoco existe se producirá un error que tratará el SML.

Opciones sobre características

En la definición de los conceptos se mencionó como se construyen incrementalmente los prototipos sin embargo se trató superficialmente cómo se vinculan con las características.

El SMO permite que tanto los prototipos como los elementos tengan un conjunto arbitrario de características y las relaciones que los vinculan se pueden calificar como *fix*, *constant* o *variable*. La diferencia entre las tres es en que momento se establece un valor a la característica.

A las relaciones *fix* se les debe adjuntar un valor en la definición del prototipo y el mismo no se podrá cambiar luego en las instancias. Es decir todas las instancias del prototipo tienen el mismo valor en esa característica.

Las características *constant* se deben establecer cuando se crean las entidades o se determina una subentidad en la definición de un prototipo pero luego tampoco podrán modificarse. Así distintas instancias podrán tener distintos valores en la característica en cuestión.

Finalmente las características calificadas como *variable* se les pueden asignar un valor durante la vida de las instancias. En el caso de las subentidades debe estar creada la entidad padre para poder asignar un valor a una característica *variable* de una subentidad.

Para las características calificadas como *constant* o *variable* es posible asignar un valor *default* en la definición del prototipo que después puede ser sobrescrito.

Además los prototipos pueden tener definidas condiciones que se evaluarán cuando se asignen valores a las características de las instancias. Se denominan *check conditions* y son expresiones booleanas sobre los valores de las características de una entidad que mientras sean verdaderas determina que la entidad es válida.

Capacidad de sensado

La capacidad de sensado del robot está fuertemente condicionada por el hardware del mismo siendo por eso que la definición más primitiva de la misma no será cubierta -por el momento- por el lenguaje de la arquitectura. Así esta información será incorporada directamente en sistema de gestión de hardware a través de un archivo de configuración o algún método similar.

Zona de sensado

En este apartado se amplía la definición de la zona de sensado ya introducida. El primer punto es la resolución en el dominio de la característica. La forma que se propone es que para cada dimensión se defina una clase de equivalencia sobre el dominio de la característica. Estas clases se pueden denotar como un rango sobre el dominio de la característica si el mismo es ordenado o sino se deberá por extensión indicar cada uno de los elementos pertenecientes a las clases de equivalencia.

Asociado a cada clase definida se incluye una estimación de la confiabilidad. La confiabilidad mide cual es la probabilidad de que en el lugar que ocupa la clase en la dimensión el valor sensado pertenezca a algunos de los valores de la clase. Un medio

para calcular una estimación de la confiabilidad podría obtenerse con el cálculo de la frecuencia relativa de una muestra suficientemente grande bajo condiciones controladas de experimentación y de hecho la definición de la función podría obtenerse autónomamente también por medio de comportamientos de experimentación. Sin embargo, para poder llevar a cabo esto es necesario ampliar la definición del lenguaje para que puedan modificarse dinámicamente las zona de sensado, hecho que se decide dejar fuera del alcance de este trabajo.
A continuación se muestras algunas ejemplos de zonas de sensado. Una cámara tiene una confiabilidad de cero en distancia pero alta en rumbo; en cambio, un telémetro láser tiene una confiabilidad muy alta en distancia, rumbo y altura pero el dominio tiene muy baja resolución ya que sólo tiene dos valores: existe y no existe. Otro ejemplo son los sensores infrarrojos que devuelven la intensidad de la señal en función del coeficiente de reflectancia IR de la superficie y de la distancia del objeto por lo tanto para obtener una mejor representación de la distancia es conveniente considerar la superficie. El modo de hacer eso es considerar una característica para cada tipo de superficie y plantear distintas regiones alcance para cada característica. Así para un mismo valor absoluto del sensor IR para superficies de alta reflectancia las zonas estarán más lejanas y para las absorbentes las zonas serán más cercanas.
Otro aspecto a remarcar es lo importante que es que las regiones de alcance sean solidarias a sus sensores. Esto se logra por medio de declarar a estas subentidades como *range* en el prototipo *self*. De modo que cuando sea preciso sensar un valor en un punto del espacio el sistema de percepción generará instantáneas prescriptivas para mover la entidad virtual de alcance. Ahí, el sistema de actuación verificará que no puede actuar sobre la entidad virtual directamente sino que debe hacerlo a través de desplazar la entidad sensor. De esta forma es como se integra el sensado activo[12].

Capacidad de actuación

Con el propósito de acotar el alcance del trabajo se decidió circunscribir por el momento la arquitectura a los robots cilíndricos con dos ruedas y dirección diferencial. De esta forma, el único modo de interacción con el ambiente es a través de sus ruedas y empujando los objetos con su superficie cilíndrica. Este límite implica también posponer un modelo mecánico genérico en cierta forma análogo al utilizado para describir la capacidad de sensado. Así, será el sistema de actuación que resuelva ad hoc la capacidad de actuación. No obstante, sí se registra -como para cualquier prototipo concreto- la definición de la superficie de contacto y de la densidad como se puede apreciar en el diagrama del modelo.

Resolución de consultas sobre prototipos

La resolución de consultas sobre la definición de los prototipos es una de las funciones principales del SMO, sin embargo, como la misma surge directamente de las capacidades brindadas por las bases de datos tradicionalmente no se describirá ya que lo más probable es que en una implementación de la arquitectura simplemente se integre un motor de bases de datos externo. Si se observa el diagrama del modelo de datos se nota que tener un lenguaje de consulta con capacidad de *path expression*[13] sería muy conveniente dado el carácter reflexivo de la definición de los prototipos.

Renderización de prejuicios

La renderización consiste en generar, a partir de la información vectorial de los prototipos, una representación de cómo sería un subconjunto de la salida de la función

[12] Si el robot implementara una funcionalidad de auto calibración entonces deberá operar sobre la definición de las zonas de sensado.
[13] Un lenguaje popular que se podría utilizar sería XPath sobre todo si se trabaja sobre un motor de bases de datos que utilice XML para describir los prototipos.

del sensor -ver sistema de gestión de hardware (SGH)-. El propósito, como ya se adelantó, es comparar el resultado de la renderización con lo obtenido del sensor en vez de analizar la salida del sensor directamente. El sistema de percepción se encarga de hacer funcionar este esquema por medio de una construcción incremental e iterativa que se aproxima a la salida completa del sensor. Eso se logra ya que el -sistema de percepción (SP)- construye *prejuicios* -se adelanta a continuación una definición de los mismos- que son validados por el sensado y en función de la validación se siguen distintos caminos que implican nuevos y diferentes prejuicios. No obstante, la construcción de los prejuicios es función del SP y se trata en la sección correspondiente.
El circuito completo comienza en el SP generando un prejuicio que es una descripción vectorial de lo que se espera sensar que es pasada al SMO. El SMO renderiza el prejuicio por medio de su motor de renderización que accede a la definición vectorial de los prototipos -incluye la definición de las zonas de sensado- generando plantillas de comparación y máscaras de sensado. Entonces, el SMO pasa las mismas al SGH para que compare y/o muestree con lo producido por el SMO con el resultado de la función de sensor correspondiente. Una vez que el SGH terminó con su trabajo -comparar y/o muestrear- informa un resultado vectorial al SP sobre los prejuicios procesados. El SP actúa en función de ese resultado creando y/o modificando prejuicios que son pasados al SMO y el ciclo se repite.

Prejuicio
Los prejuicios son artefactos del sistema de percepción que utiliza para describir lo que se esperaría sensar. Desde el punto de vista del SMO un prejuicio es una estructura de elementos de prototipos de uno o más prototipos concretos organizados espacialmente que se esperan captar en una región del espacio particular llamada espacio incógnito. Además, los elementos seleccionados por el SP para conformar un prejuicio tienen que tener la misma característica. Las posiciones de los elementos de los prototipos utilizados en un prejuicio quedan determinadas por la posición del elemento en el prototipo del cual forma parte. Por lo tanto, es la posición de los centros de los prototipos utilizados en el prejuicio los que deben declararse en el mismo y la referencia de la posición de los centros de los prototipos de los prejuicios es con respecto a la entidad virtual espacio incógnito.

Motor de renderización
Está fuera del alcance de la arquitectura la especificación del motor de renderizado ya que es un campo sumamente estudiado. Se presume utilizar las tecnologías de renderización en tiempo real para la construcción del motor de renderización e inclusive podría ser un componente externo en una implementación de la arquitectura. De ahí, que el tema de cómo el motor de renderización construye la “imagen” es dejado de lado en este trabajo. Es claro que a medida que el SP y el motor de renderizado sean más sofisticados se requerirá un ambiente para el robot menos estructurado y que en un ambiente sumamente estructurado el motor de renderizado será relativamente sencillo de construir.

Paralelización de configuraciones de hardware
Este servicio consiste en informar si es posible que dos configuraciones de hardware sean establecidas al mismo tiempo, en paralelo. Es utilizado por el sistema de percepción para confeccionar la estrategia de sensado. Por ejemplo, si se ajusta un sensor para que de una alta respuesta al sensar un determinado valor no se puede al mismo tiempo obtener una alta respuesta para otro valor.

Firmas de prototipos

Este servicio brinda una firma o huella de un prototipo, esto es, un conjunto de tuplas de identificadores de elemento y de valor de dominio que son muy selectivos para el prototipo y la característica recibidos con respecto a todos los prototipos almacenados de forma que lo destaque perceptivamente sobre todos los demás que estén registrados en el SMO. También se agrega a los parámetros recibidos para disminuir el espacio de posibilidades el tipo de selección -prejuicio positivo o negativo (ver Prejuicio Pág. 101)- y una orientación espacial. Por medio de Bayes[14] se puede precalcular el resultado cada vez que se almacena un nuevo prototipo.

[14] Se refiere al teorema de Bayes de probabilidad condicional.

Sistema de gestión del hardware (SGH)

Objetivo

Gestionar el uso de las entidades propias del robot del tipo actuador y sensor para que los demás sistemas no tengan que considerar las particularidades del hardware.

Introducción al SGH

El objetivo del SGH es el de aislar las particularidades del hardware de los actuadores y de los sensores al resto de los sistemas [2][5][6]. Se lleva a cabo a través de una interfaz que se sustenta en las características físicas, en las dimensiones del espacio y en las configuraciones de hardware. Así, el sensado consiste en cómo obtener qué valor tiene una característica en una posición del espacio. La actuación por el momento será realizada principalmente por el sistema de actuación hasta que se desarrolle la descripción general de la mecánica de los robots en la arquitectura. De modo que el SGH se limitará a ejecutar los comandos que le envíe el SA.

Para cumplir con su objetivo el SGH realiza las siguientes funciones:

- Informar el estado de las entidades propias que se obtiene de los sensores de propiocepción.
- Emparejar los datos sensados con las plantillas de comparación de los prejuicios.
- Obtener de los sensores los valores de las características físicas variables con las máscaras de obtención de los prejuicios.
- Convertir los comandos en instrucciones para los drivers del hardware.

Definición de conceptos

Driver

Los drivers son el límite inferior en el nivel de abstracción del alcance cubierto por la arquitectura. Se supone que los drivers se utilizan ajustando parámetros de valores atómicos y que el control de la ejecución de los mismos se lleva a cabo con alguna primitiva que dispone la arquitectura para ese fin. Los drivers deberán ser incorporados al robot por el diseñador de manera tal que puedan ser controlados por la anterior primitiva citada.
No obstante, como ya se mencionó en la sección del SMO, por el momento los drivers no utilizarán parámetros sino que las diferentes configuraciones de los mismos deberán ser ajustadas por el diseñador e incluidas al robot como drivers distintos. Esto es así para recortar el alcance del presente trabajo excluyendo el protocolo de comunicación de descripción y utilización de drivers entre el SGH y el SMO.

Función sensor

La función sensor es un mapeo que debe hacerse con la salida sensada de los driver para adecuarlos al esquema espacial y de características manejados en la arquitectura. Cada zona de sensado registra cual es la función sensor que debe utilizar y los parámetros de la función tienen como fin seleccionar desde el *buffer* del sensor el resultado de interés. Así, los parámetros son las dimensiones espaciales -cada una con un rango escalar- y un conjunto de características y el resultado de la función es un conjunto de tuplas de los valores sensados en su contexto espacial. Específicamente

cada tupla está conformada por un escalar para cada dimensión, una característica y un valor de característica.
Los rangos escalares de las dimensiones pueden estar vacíos en la invocación de la función y en ese caso la función debe devolver el conjunto de tuplas para cubrir todo el rango cubierto por el sensor en esa dimensión. En cambio las dimensiones no se pueden omitir ya que sino habría que proveer una función de agregación para agrupar los valores que se superpondrían por la ausencia de una o más dimensiones.

Propiocepción

Los sensores de propiocepción son sensores que miden el estado de las entidades propias y el fin de los mismos es brindar directamente un feedback en tiempo real al SA. Para satisfacer este requisito sacrifican la calidad para obtener un tiempo adecuado de cómputo. Sin embargo, el SP aún puede utilizar estos sensores de todas formas y entregar una respuesta integrada de mayor calidad.
Estos sensores se modelan en el SMO igual que el resto de los sensores. Pero, como un sensor en general no obtiene la posición de la entidad objetivo sino, que obtiene la posición de ciertos valores de una característica en algunas dimensiones -y es el SP el que se encarga de obtener la posición de la entidad objetivo en función de los datos del sensor- es que los sensores de propiocepción requieren ciertas restricciones en su función sensor para poder informar la posición de la entidad objetivo -i.e. una propia- sin necesidad de que medie el SP.
La restricción implica que la característica para el codominio de la función sensor debe ser compatible con un tipo booleano con el significado si la entidad objetivo está presente o no en el espacio. Pero como la entidad objetivo es una propia entonces la función sensor siempre devolverá verdadero en la característica y la posición de “ese” valor será la posición de la entidad propia sensada por propiocepción. Así se logra que el sensor de propiocepción informe directamente la posición de la entidad objetivo sensada. Tal como los demás sensores, los de propiocepción, pueden tener más de una característica y, en este caso, se puede utilizar para obtener las funciones derivadas de la posición -i.e. velocidad, aceleración y fuerza- estableciendo una característica por dimensión y función derivada.

Descripción general

A continuación se describirán las cuatro funciones que realiza el SGH.

Procesamiento de la propiocepción

El SGH comenzará a proveer información de propiocepción cuando reciba un mensaje *pedido de propiocepción* desde el sistema de actuación -SA-. Los atributos del mensaje son: zona de sensado, frecuencia máxima de refresco, prioridad y frecuencia mínima de refresco. La frecuencia máxima es para evitar que el SGH moleste demasiado al SA con actualizaciones de información de propiocepción poco relevantes. La prioridad surge de la instantánea prescriptiva que esté procesando el SA que necesite de la información del sensor de propiocepción. La frecuencia mínima es una forma general de tolerancia temporal que el SA obtiene también de la instantánea prescriptiva.
El SGH lo primero que lleva a cabo es la verificación en su estado interno si lo solicitado en el pedido de propiocepción ya está disponible en el SGH sino no es así entonces, con la información de la zona de sensado recibida, debe ajustar el hardware.
Luego establece el mecanismo de envío de la respuesta de modo que satisfaga la frecuencia mínima y máxima definida, de forma tal, que el SGH llamará a la función driver por lo menos a la frecuencia mínima y, de acuerdo a la carga del sistema, incrementará la frecuencia hasta la frecuencia máxima.

Finalmente la respuesta a un pedido de propiocepción se realiza enviando el mensaje *propiocepción directa*. Los mismos tienen los siguientes atributos: identificador del sensor de propiocepción y un conjunto formado de tuplas de este tipo (dimensión, coordenada, característica, valor).

Emparejamiento de prejuicios

Cuando el SGH recibe de parte del SMO un mensaje de plantilla de comparación lo primero que hace es analizar si el SGH ya tiene ajustado el sensor y configuración de hardware solicitados. Eso lo lleva a cabo comparando sus registros internos con la información de la zona de sensado que viene en el mensaje. De no ser así procede a configurar el hardware para poder sensar lo solicitado en la plantilla con la información de la zona de sensado que provino en el mensaje. La prioridad y la tolerancia temporal, también incluidos en el mensaje, son utilizadas para determinar si efectivamente podrá usar o no el hardware en cuestión.

Una vez que el SGH alcanzó la capacidad de sensar, procesa la salida de la función sensor para comparar las tuplas de la plantilla con la tupla que le corresponde incluidas en el mensaje. Este procesamiento incluye un análisis estadístico para el grado de emparejamiento entre datos sensados y la información de la plantilla.

Realizadas estas tareas envía su respuesta al sistema de percepción con un mensaje de emparejamiento. Dicho mensaje contiene el grado de confiabilidad de reconocimiento determinado para el prejuicio. En caso que no se halla podido usar el hardware informa al SP un error de sensado.

Obtención de valores de características

El procesamiento del mensaje *máscara de obtención* es similar al del mensaje de *plantilla de comparación*. Así, cuando se recibe un mensaje de *máscara de obtención* lo primero que se verifica es la disponibilidad del hardware.

La diferencia se establece a partir del momento en que se puede sensar. En este caso se procesa el resultado de la función sensor para el espacio cubierto por cada variable declarada. Similarmente se analiza estadísticamente la muestra por cada variable pero en vez de determinar un grado de confiabilidad de emparejamiento se determina estadísticamente un valor representante para cada una de las variables junto con la confiabilidad de que ese valor sea realmente significativo para toda la variable en cuestión.

Una vez procesadas todas las variables el final es también muy similar al mensaje de *platilla de comparación*. El SGH envía un mensaje *valores de variables* al sistema de percepción -SP- informando el valor para cada variable o un error si no pudo llevarlo a cabo.

Ejecución de comando de actuadores

Tal como se mencionó antes la presente descripción de la arquitectura omite una descripción genérica de la actuación de los robots y en cambio se circunscribe a un robot cilíndrico con dirección diferencial con dos ruedas. Dada esta reducción del alcance la arquitectura sólo constituirá un par de mensajes. El primero, *comando actuador -solicitud-*, desde el sistema de actuación hacia el SGH y la respuesta del estado del mismo, *comando actuador -respuesta-*, desde el SGH hacia el SA. El mensaje comando enviará los típicos comandos para el movimiento de este tipo de robot.

Sistema de motorización del lenguaje (SML)

Objetivo

El objetivo del sistema de motorización del lenguaje es brindar el medio para que un programa escrito en el lenguaje de la arquitectura pueda ser ejecutado.

Introducción al SML

El propósito de plantear la conveniencia de un lenguaje en la arquitectura es facilitar el desarrollo de aplicaciones para robots que puedan adaptarse en forma dinámica a variaciones del hardware necesario para ejecutarlas.
En general, el software de alto nivel de un robot es un componente complejo que suele ser muy dependiente del hardware y esto hace engorroso ofrecer distintos modos de cómo llevar a cabo una tarea del robot porque habría que programar completamente cada uno de ellos. No obstante, para un robot autónomo, es altamente deseable que existan modos alternativos de desenvolvimiento para una misma tarea porque pueden ocurrir situaciones imponderables que serían subsanables con la utilización del hardware de un modo distinto al que originalmente fuera pensado.
De manera que, para contar con la posibilidad de esos modos alternativos, la arquitectura propone separar en los comportamientos del robot la declaración de lo que estos deben hacer de la implementación de cómo deben hacerlo. Se alcanza este objetivo con un lenguaje que permita expresar los comportamientos sin referencia alguna a sensores o actuadores específicos -i.e declarativamente- y con un conjunto de sistemas que se encarguen de transformar las descripciones declarativas provenientes del lenguaje en las selecciones y activaciones de sensores y actuadores que, efectivamente, lleven a cabo los comportamientos.
En esta sección se describirá el lenguaje propuesto para la arquitectura. Se trata de un lenguaje del tipo 4GL es decir, orientado a la confección de programas acotados a un dominio particular por medio de la utilización de elementos propios del dominio de aplicación en su gramática y semántica lo que le permite así presentar un carácter declarativo y un alto nivel de abstracción. En el caso de este lenguaje el carácter declarativo se observa en que en la programación de la aplicación los comportamientos se especifican por la declaración de la posición que ocupan, podrían ocupar y deben ocupar las entidades que representan el mundo. Por el lado de la característica de alto nivel de abstracción se refiere a que las estructuras y objetos programáticos utilizados en el lenguaje son representaciones cercanas a las existentes en el mundo físico; de modo, que deberían ser mínimos los elementos programáticos típicos de los lenguajes de computación -como variables y ciclos- necesarios para representar el mundo de la aplicación que estén más allá de lo alcanzado por el lenguaje.
Constitutivamente los programas de este lenguaje se caracterizan por estar compuestos por bloques de código que describen a los prototipos -ver sistema de modelización de objetos SMO- y por bloques de código que denominados **evoluciones** que describen algorítmicamente un determinado movimiento aplicable a un prototipo. Así, en el lenguaje de la arquitectura, un programa estará constituido por prototipos y evoluciones. A continuación se presentan unos conceptos necesarios para la descripción del lenguaje.

Prototipo

Para el lenguaje los prototipos representan las clases de objetos que **podrían existir** en el mundo de la aplicación del robot. Los prototipos son unos de los elementos

centrales del sistema de modelización de objetos y es por ello que estos son descriptos en aquella sección. Sin embargo, los prototipos son construidos por medio del lenguaje y es a través de la ejecución del programa que las definiciones de los mismos alcanzan al SMO. En el apéndice sobre la gramática se ven las estructuras del lenguaje que posibilidad su construcción.

Evolución

Las evoluciones definen como se deben "mover" o evolucionar los prototipos por el espacio. Esta definición es declarativa con el fin que sean los sistemas de la arquitectura los que automáticamente se encarguen de seleccionar los sensores y actuadores para llevar a cabo el "movimiento". Por lo tanto, el trabajo de programación será más declarativo, y por ende más sencillo, cuanto más versátiles sean los sistemas de la arquitectura para convertir la declaración en implementación.
Sin embargo, descripciones procedimentales también son posibles y la elección entre un estilo declarativo o procedimental depende de lo complejo del movimiento y de las capacidades algorítmicas y de procesamiento de los sistemas. El estilo procedimental es posible porque el robot está compuesto por entidades y, si se declaran movimientos para los prototipos de las entidades propias, entonces, cuando se active luego la implementación automática se cumplirá con la descripción declarativa pero, por la forma en que fue hecha la misma, implicará un procedimiento. Así, está abierta la posibilidad para establecer que el comportamiento del robot sea procedimental en vez de declarativo cuando se necesite.
Además, es también una función de las evoluciones expresar la coordinación de otras evoluciones para cuando aquellas se estén ejecutando. Así mismo, como los comportamientos del robot, la coordinación también se expresa declarativamente.

Entidad

Las instancias de los prototipos son llamadas entidades y las mismas representan los objetos que **existen** en el mundo de la aplicación del robot. Además son los elementos centrales del sistema de integración espacio temporal en cuya sección se explican completamente. Las entidades, como concepto de programación, son variables globales compartidas por todas las instancias de las evoluciones.

Recorrido

Los recorridos son las instancias de las evoluciones. Una evolución puede ser instanciada en varios recorridos. Los recorridos son declarados en el código de las evoluciones en dónde se determina a partir de que evolución se instanciará. La excepción a esta regla es el recorrido *Main* que no puede tener más que una única instancia y que, además, no puede ser declarada en ninguna evolución ya que la creación de la instancia en este caso depende exclusivamente del motor del lenguaje.

Instantánea

Una instantánea contiene los datos de una entidad en un momento determinado ya que el modelo del mundo pueden registrarse varios momentos consecutivos de una entidad; así, en este aspecto, una instantánea es como el fotograma de una película. Los datos contenidos en las instantáneas son la posición de la entidad -i.e. el valor del escalar en cada una de las dimensiones espaciales- y los valores de las características variables -visto en la sección SMO-.

Propósito de uso

Cuando una entidad es usada en un evolución se presenta el hecho que puede ser utilizada con diferentes propósitos. Estos propósitos determinan indicios que los sistemas que realizan la transformación de la declaración a su implementación utilizan para determinar la transformaciones. Es por eso que se estableció que el lenguaje

considere explícitamente tres tipos de propósitos llamados **descriptivo**, **prescriptivo** y **especulativo**. Puesto que los datos de las entidades se contienen en las instantáneas es que se define que por cada entidad se asocie tres secuencias de instantáneas ordenadas temporalmente, una para cada tipo de propósito.
El descriptivo tiene como fin describir el mundo del robot, tanto para el pasado, presente y futuro -predicción-. El prescriptivo representa lo que el robot necesita que se cambie del mundo -siempre es en el futuro- y finalmente el especulativo describe cómo podría ser el mundo y se sustenta en el conocimiento que se tenga sobre el dominio de aplicación del robot, teóricamente podría ser para el pasado, presente y futuro pero en la práctica se elimina el pasado. Por ejemplo, en una aplicación para jugar un deporte, con una instantánea especulativa se podría decir que al comienzo del juego la pelota estará en el centro de la cancha.
En general es imprescindible que el robot actúe sobre las cosas del mundo pero para eso necesita primero saber dónde estas están. Sin embargo, antes de poder sensar la cosa es posible saber dónde podría encontrarse dicho objeto por conocimiento derivado del dominio de aplicación. Así, previamente a sensar, un conjunto de instantáneas especulativas representarán el conocimiento del dominio de aplicación de dónde podrían estar los objetos de interés, luego sobre dichas posiciones especulativas se sensará y es posible que los objetos de interés sean sensados en una de aquellas posiciones. En ese momento, el hecho de sensar un objeto se representa con una instantánea descriptiva para que, finalmente con esa información, plantear hacia dónde deben moverse los objetos de interés a través de instantáneas prescriptivas.
Por último las instantáneas prescriptivas serán procesadas por los sistemas de la arquitectura y estos activaran los actuadores con el fin de cumplir con la prescripción. Cabe aclarar que el sistema de la arquitectura que se encarga de las instantáneas descriptivas pueden necesitar “mover los sensores” y por lo tanto para realizar el sensado activo generará también instantáneas prescriptivas para los sensores. De ese modo se cumple con el fin de la aplicación del robot.

Tipo de evolución

El modelo del mundo que la arquitectura propone se centra en las entidades y cómo estas se mueven o evolucionan en el tiempo. Por eso es que se tipifican a las evoluciones en: las que describen lo que podría ser, aquellas que describen lo que es y será y finalmente las que describen lo que se necesita que sea. Respectivamente nos referimos a las especulativas, descriptivas y prescriptivas.
Constitutivamente los tres tipos son idénticos y sólo cambia el significado que le damos a sus instantáneas generadas. En base a los diferentes significados es que las evoluciones de una clase sólo pueden utilizar a otras evoluciones de la misma clase y de ahí que la evolución *main* deba ser prescriptiva.
Las evoluciones descriptivas y especulativas son utilizadas solamente por el sistema de integración espacio temporal -SIET- y el nombre de las mismas debe ser igual al del prototipo para el cual generan los datos de modo que el SIET pueda hallarlas cuando necesita los datos descriptivos o especulativos de las entidades de un prototipo.

El motor de ejecución del lenguaje

Si bien la presente sección se refiere al sistema de motorización del lenguaje en la misma se detallan principalmente los elementos estructurales y semánticos del lenguaje. Especialmente se describen las estructuras de control para la implementación del mecanismo de sincronización, sin intentar una descripción profunda del motor del lenguaje. Se relegan, de este modo, cuestiones importantes y habituales en la implementación de un lenguaje, como podrían ser la gestión de memoria, la

representación interna del código o la ejecución en sí, ya que no resultan imprescindibles en esta descripción orientada a la robótica.
Un programa del lenguaje se conforma de prototipos y evoluciones. El mismo es ejecutado por el motor de ejecución cuya finalidad es intercambiar mensajes con los sistemas de integración espacio temporal y de modelización de objetos -SIET y SMO- de acuerdo a lo expresado en el programa para que, a través de estos sistemas, el robot lleve a cabo el cometido de la aplicación.
Todo programa de este lenguaje, de modo semejante a otros lenguajes, debe tener punto de inicio formalmente identificable, siendo en este caso una evolución con el nombre reservado de *main*. Así, cuando se inicia el motor del lenguaje, este toma el código del programa y lo procesa a partir de *main.* Una de las funciones principales del motor es la creación de las instancias para las evoluciones y los prototipos, esto es los recorridos y las entidades.
La creación de las instancias de las evoluciones, es decir, los recorridos, han de estar siempre explícitamente expresadas en el código con instrucciones, sin embargo, *main* es un caso particular ya que en ningún lugar del código se encuentra explícitamente la creación del recorrido *Main*, lo que significa que *Main* es transparentemente creado tras el *reset* y, por lo tanto, será el primero en ejecutarse. Es un supuesto de este lenguaje que una misma evolución pueda tener varios recorridos concurrentes para realizar diferentes acciones, sin embargo, otro aspecto particular de *main* es que sólo puede admitir uno y sólo un recorrido. Así como los recorridos se crean expresando este hecho explícitamente en el código, al destruirse también deben expresarse de modo análogo. Por el contrario, y a diferencia de los recorridos que se deben declarar, crear y destruir explícitamente, las entidades del prototipo sólo se declaran en las evoluciones e implícitamente se crean y destruyen en concordancia con la creación y destrucción de los recorridos que las utilizan.
La creación de una entidad acontece cuando es utilizada por primera vez por un recorrido y su destrucción se llevará a cabo automática e implícitamente, al destruirse el último recorrido que la utilice.
Es importante que el motor del lenguaje no solicite las entidades al SIET apenas se inicia el recorrido sino cuando las mismas se usen o con algún otro modo de perezoso -*lazy*-. Esto se define así por la evoluciones que se encargan de hacer el mapa ya que sino la creación de sólo una entidad provocaría una reacción en cadena creando todo el resto del mapa.
Si un prototipo no tiene una instancia entonces la misma se crea pero si ya existe la instancia *necesitada* del prototipo entonces se utiliza esa en vez de crear otra. Esto significa que un prototipo puede tener varias entidades asociadas y las entidades son globales para todos los recorridos.
Además, cuando se crea una instancia para una entidad, se habilita al SIET de crear un recorrido para dicha entidad que se llamará igual que su entidad y cuya evolución se también se deberá llamar igual que el prototipo correspondiente. Por ejemplo, supóngase un prototipo pelota, entonces se puede escribir una evolución descriptiva pelota que sea el predictor para el prototipo. De modo que cuando se cree la instancia de pelota PelotaRoja se habilite la posibilidad al SIET de crear el recorrido descriptivo PelotaRoja a partir de la evolución pelota. En caso de crear otra instancia -PelotaAzul por ejemplo- para pelota lo propio pasará con su recorrido descriptivo. Esto es así ya que el estado de cada recorrido es diferente ya que pertenecen a distintas entidades. La descripción es totalmente análoga para las evoluciones especulativas.
El escenario completo de uso es así. Un recorrido prescriptivo necesita una instantánea descriptiva de una entidad por lo el SML envía un pedido al SIET para que le provea dicha información. Si el SIET no puede suministrarla le pide al sistema de percepción - SP - que ubique la entidad solicitada; para eso el SP puede decidir, para llevar a cabo el pedido, basarse en la información del dominio de aplicación que le brinda la

evolución especulativa. Así hace un pedido al SIET que provoca que cree el recorrido especulativo obteniendo instantáneas especulativas que pasan al SIET y luego al SP para ayudarle a sensar. Una vez que el SP obtiene los datos para la instantánea descriptiva se los pasa al SIET que se los envía a su vez al recorrido prescriptivo que hizo la solicitud original como así también destruye el recorrido especulativo recién recreado. Una vez que el recorrido prescriptivo original obtuvo la información del presente podría necesitar también información del futuro de la entidad que el SIET resuelve creando un recorrido descriptivo para la entidad que, en base a la información de la instantánea descriptiva presente, genera las instantáneas descriptivas del futuro. Por último, cuando cesan los pedidos de futuro para la entidad, el SIET destruye el recorrido descriptivo asociado.

El lenguaje

De lo que resta de la sección del SML se describirá en detalle las estructuras internas que organizan las evoluciones mientras que las declaraciones de los prototipos serán sólo tratadas en el apéndice “Gramática del lenguaje”.
Por ser un lenguaje orientado a aplicaciones de robótica el mismo incluye que las variables y las expresiones además de ser de un tipo de datos sean también de una unidad física determinada. El fin de las esta característica del lenguaje es disminuir la frecuencia con la cual se presentan errores en los robots debido a una inadecuada asignación de unidades [1][3][6].

Secciones de una evolución

El código de las evoluciones se dividen en cinco secciones - *state*, *context*, *requeriment*, *produce* y *syncronization* - con funciones precisas que a continuación se detallan.

Sección state

En la sección *state* se declaran las variables auxiliares para que los recorridos puedan mantener un estado privado para el algoritmo que estén computando. Esta sección es opcional y las variables declaradas en esta tienen como ámbito de alcance toda la evolución. Esto es así porque este estado tiene también como propósito ser el medio no estructurado del lenguaje para comunicar entre sí las distintas secciones en caso de ser necesario. El estado acompaña la existencia del recorrido, esto es, se inicializa con la creación del recorrido y se destruye junto con el recorrido.

Sección context

En *context* se declaran las instancias que utilizarán los recorridos que se instancien de la evolución. A a cada instancia declarada se le asigna un nombre local aunque externamente cada una tiene un identificador global. El propósito es establecer un único punto en donde se pueda ver claramente con qué elementos externos a la evolución la misma trabajará y a la vez que se les suministra un nombre local con significado para el propósito de la evolución.
Las evoluciones describen a las entidades que necesitan con un predicado. El espacio sobre el cual opera aquel predicado es la unión de las definiciones de los prototipos, el estado de las entidades en el momento de evaluar el predicado –por ejemplo la posición de las entidades- y siempre se intenta reutilizar un entidad que construir otra nueva. Por ejemplo, si una evolución necesita un perno específico, sólo le servirá ese perno. En cambio si una evolución necesita un proyectil, el perno o cualquier otro objeto de cierto tamaño y masa le servirá. Que objeto se seleccione sólo depende de como se describa en el predicado. De ahí, que a pesar que el sistema de base de datos del robot puede tener muchos prototipos se espera que en todo momento se hayan creado instancias sólo para un subconjunto reducidos de los mismos.

Declaraciones de instancias

Hay cuatro tipos de declaraciones:

- **recorridos**: Los recorridos son útiles ya que permiten especificar el comportamiento de las evoluciones de un modo más económico por medio del efecto que producen los recorridos de otras evoluciones. En cada declaración se asocia un nombre local para el recorrido subordinado. Los recorridos no son reutilizables a diferencia de las entidades por eso un recorrido sólo será instancia de un único recorrido padre. Se considera que la importancia relativa entre los recorridos subordinados para el recorrido de la evolución coincide con el orden de estas declaraciones.
- **protagonista**: Son las declaraciones de las entidades que son la razón de ser de la evolución. La declaración consiste en una asociación de un nombre local con una expresión de entidad. La expresión de entidad es un predicado que opera sobre las definiciones de los prototipos y sobre las entidades presentes en el modelo y que debe devolver una única entidad. Cuando la expresión retorne una entidad existente, será reutilizada; si en cambio no existe ya en el modelo, se creará una nueva. Luego para ambos casos casi siempre será necesario -si la entidad es concreta- que otros sistemas previamente sensen a la entidad para que el recorrido pueda utilizarla finalmente. Las expresiones son evaluadas cuando un recorrido de la evolución se instancia. En caso que la expresión de entidad no retorne una y sólo una entidad se provoca un error que activa el bloque *resource* - perteneciente a la sección *produce* - del recorrido llamador. No obstante el sublenguaje de las expresiones de entidad tiene medios para garantizar devolver sólo una.
- **reparto**: El reparto también declara entidades para los recorridos que se instancien de una evolución sin embargo las expresiones de entidad son evaluadas en cada ciclo del bloque *while* de la sección *produce*. Esto permite que cada expresión defina un rol necesario para la evolución que podrá ser llevado a cabo por cualquier entidad que la cumpla permitiendo así cambiar dinámicamente estas entidades en el transcurso de la vida de un recorrido. Al igual que con las protagonistas cada expresión debe devolver una única entidad y la resolución de este asunto es idéntico a cómo se resolvió con aquellas.
- **invitada**: Las invitadas son entidades seleccionadas por un recorrido padre para pasar a un hijo. En esta declaración se define una expresión de entidad para cada invitado que los mismos deben satisfacer para poder ser aceptados por los recorridos instanciados de la evolución. Un recorrido padre puede seleccionar como invitado para un hijo a un protagonista, una entidad del reparto o un invitado de sí mismo. Además de seleccionar la entidad el padre se mantiene vinculado con las entidades que eligió para invitados de sus hijos por medio de los requerimientos que defina para aquellos. De esta forma los recorridos ancestros pueden controlar el matiz de actuación de los recorridos descendientes. Las expresiones de entidad de los invitados son evaluadas cuando se crea el recorrido si provienen de los protagonistas y en cada ciclo *while* si lo hace del reparto. Sin embargo, los requerimientos de los invitados independientemente del origen son siempre evaluados en el ciclo while ya que eso permite los ancestros cambien los mismos y que se vea reflejado en los descendientes. Finalmente las evoluciones preparadas para recibir invitados también pueden aumentar las restricciones de los requerimientos de modo de seguir cumpliendo con los que provengan con los invitados pero refinando el actuar de los descendientes. En caso que la entidad suministrada no cumpla con la restricción provoca un error que activa el bloque *resource.*

Entidad de referencia

Las entidades de referencia son entidades declaradas en la sección *context* que son utilizadas en la evolución -en particular en las en las secciones *produce* y *syncronization*- como puntos de referencia espacial para denotar la posición de las demás entidades. Dado, este propósito de referencia, las entidades asignadas para cubrir esta función no pueden ser abstractas ya que las mismas carecen de las dimensiones espaciales. Si el recorrido tiene inconvenientes con las instantáneas descriptivas de las entidades de referencia entonces informa un error de recursos y el recorrido no puede continuar hasta que se subsane este problema. Las entidades para las cuales se generan instantáneas no pueden ser declaradas como la entidad de referencia de *context* ya que no habría un modo de expresar la variación del movimiento con respecto a la misma. En el caso particular de las evoluciones descriptivas y especulativas puede ser común que deban tener como entidad de referencia a una que pertenezca a los invitados.

Sección requeriment

El lenguaje tiene una impronta declarativa que relega cómo son llevadas a cabo las acciones delegándolas en los sistemas de la arquitectura. Sin embargo, es necesario mantener el control del matiz de cómo el robot lleva a cabo sus acciones y el modo de especificarlo es mediante requerimientos asociados a las dimensiones de las instantáneas. Es por eso que las instantáneas tienen campos para metadatos para señalar los requerimientos que se necesitan, permitiendo por ejemplo, tener mucha precisión y confiabilidad en el rumbo pero poca en la distancia. Esto afecta tanto al sensado como la actuación del robot.

Por el lado de los sensores, si un recorrido necesita una entidad la necesidad de resolución y confiabilidad de la posición de la entidad no es la misma si la misma está lejos o si está en contacto con el robot. Además, en general distintas resoluciones y/o confiabilidades significan distintos costos de obtención, por lo tanto, el plantear las necesidades justas posibilita una mayor eficiencia en los recursos del robot.

El propósito de los requerimientos es contribuir a la flexibilidad de la arquitectura al separar la descripción del movimiento de un prototipo del modo o estilo de la actuación para llevarlo a cabo. Por ejemplo una evolución llamada rodear podría describir cómo una entidad rodea a otra. Supóngase que el ambiente es una obra en construcción, dónde el tiempo es dinero, y la entidad que se mueve es el robot y la entidad rodeada es en un caso un agujero en una losa para que pase una chimenea y en otro un montón de arena; entonces, las consecuencias de actuar rápido son muy distintas entre uno y otro escenario. En el primero se espera que se actúe despacio mientras que en la segunda que lo haga rápido. Sin embargo, acá la velocidad no es un requerimiento sino una consecuencia del verdadero requerimiento que es la confiabilidad que se necesita tener en las distancia entre el objeto rodeado y el robot. Luego, los sistemas de la arquitectura, lo resolverán modificando la velocidad de actuación.

Requerimiento

Como se vio en la introducción existen tres usos en que una entidad puede ser usada por una evolución -i.e. especulativo, descriptivo y prescriptivo-. Puesto que las instantáneas son para una entidad determinada toman de esta el tipo de entidad clasificándose por tanto en: concretas, abstractas y virtuales. Si bien de esto se deduce que existan nueve posibles tipos de instantáneas en la práctica sólo se presentan cuatro -en el apéndice *Estructura de las instantáneas* se describen la conformación de los distintos tipos de instantáneas-.

Los requerimientos no son otra cosa que los atributos de las instantáneas cuyos nombres finalizan con *req*[15]. Los requerimientos se definen para las entidades declaradas en la sección contexto sin distinguir que sean protagonistas, del reparto o invitadas.

Rasgo

Para una misma entidad pueden definirse distintos grupos de requerimientos ya que una entidad puede ser usada en diversas circunstancias. Se denomina *rasgo* al grupo de requerimientos de una entidad ajustados para considerar una circunstancia dada. Así, de lo dicho se desprende: un rasgo es para una única entidad, una entidad puede tener muchos rasgos y un rasgo tiene varios requerimientos.

Estilo

Un estilo es una agrupación de rasgos para distintas entidades cuya función es expresar cómo se llevará a cabo el movimiento descripto por la evolución en el escenario. A los estilos se les asocia un nombre para poder hace referencia a estos en la sección *produce* de la evolución.

Toda evolución tiene que señalar cuál de sus estilos será el inicial, hecho que se hará indicándolo con un calificador *primer* junto al nombre del estilo. Se cambia de estilo con la instrucción *changestyle*.

Una vez establecido un estilo se les asigna a las entidades involucradas en el mismo los valores determinados en los requerimientos. De ese modo los pedidos de entidades y las instantáneas a generarse empezarán a considerar los nuevos requerimientos. Así mismo, en caso de usar recorridos que utilizan entidades invitadas los requerimientos también serán enviados a los recorridos subordinados para que estos sean aplicados también allí.

Sección produce

Una evolución puede o bien definir cómo coordinar la generación de las instantáneas de sus recorridos descendientes, definir cómo generar sus propias instantáneas o realizar ambas cosas. Cualquiera de estas actividades que realice es llevada a cabo en el sección *produce*. La misma se divide en los bloques: *start*, *while*, *finish* y *resource* que más adelante se describen.

No obstante, cual sea el modo de generar instantáneas, cinco parámetros controlan la generación de las mismas:

- Velocidad de reacción: es la frecuencia de cada cuanto se debe generar una instantánea reactiva.
- Velocidad de planificación: es la frecuencia de cada cuanto se debe generar instantáneas que pertenezcan a un tren de instantáneas formada por una reactiva y el resto de instantáneas planificadas -pudiendo no haber ninguna de estas últimas-. Esta frecuencia tiene que ser mayor o igual a la establecida en la velocidad de reacción y comensurable de modo que las instantáneas reactivas coincidan en el tiempo con lo indicado por el parámetro de velocidad de reacción. Por eso se denota con un número natural con cero.
- Horizonte: que determina la duración máxima del plan.
- Delta *time*: porcentaje del tiempo que media entre dos instantáneas consecutivas según lo expresado en la velocidad de reacción. El delta *time* se utiliza para construir el rango de *time* -ver SIET- de modo de poder alinear aproximadamente las instantáneas de los diferentes recorridos en el futuro. Se aplica en la definición el término *aproximadamente* ya que la sincronización

[15] No obstante, en el momento de escribir el código del lenguaje no se deberá escribir la partícula *req* ya que quedará determinada por el contexto de escritura.

es asincrónica y la alineación definitiva dependerá de las condiciones de sincronización.

- Delta *processtime*: porcentaje del tiempo que media entre dos instantáneas consecutivas según lo expresado en la velocidad de planificación. El delta *processtime* se utiliza también para construir el rango de *processtime* -ver SIET- que determinar la validez para las instantáneas generadas. Así antes del límite inferior la instantánea no puede ser materializada y más allá del límite superior del rango debe cancelarse porque se ha vencido.

Si se ajusta el horizonte a cero se inhibe la planificación y se obtiene un recorrido 100% reactivo. Puede suceder que si hay escasa capacidad de procesamiento no se alcance a cubrir todo el horizonte acortando implícitamente el horizonte y manteniendo como prioridad respetar la velocidad de reacción.
Las instantáneas tiene dos atributos que son privados y gestionados automáticamente por el lenguaje: el identificador de la instantánea y si las misma es reactiva o planificada. Ambos atributos se usan para seleccionar de forma eficiente las instantáneas planificadas que serán "corregidas" ya que en cada iteración de *while* se actualiza el plan en base a la nueva información del presente que se dispone.
El *time* y el *processtime* también son usados para hacer transparentemente el pedido de instantáneas descriptivas del futuro para llevar a cabo la planificación y así obtener durante el rango *processtime* los datos de las instantáneas descriptivas que pertenezcan al rango *time*.
Los comandos al igual que las instantáneas son o bien reactivos o bien planificados y disponen de los correspondientes rangos de tiempos de *time* y *processtime*.
Si un recorrido se arranca y el tiempo del comando *start* es el futuro entonces este recorrido generará instantáneas planificadas hasta que el presente alcance el momento indicado en el *start* y ahí empezará a realizar también las instantáneas reactivas.
Análogamente sucede con el *stop*. Si su *time* está en el futuro comienza deteniendo la planificación de instantáneas a partir de ese instante para sólo destruir el recorrido cuando alcanza el presente. Los comandos toman su tiempo de igual modo que las instantáneas generadas.

Bloque while

En el bloque *while* es donde se generan las instantáneas ya sea por medio del subloque *perform* o por la activación y desactivación de recorridos subordinados. El bloque *while* es implícitamente un ciclo que perdura mientras esté activo el recorrido. El primer ciclo se ejecuta después del bloque *start* y el último ciclo será el anterior a la ejecución del bloque *finish*.
Durante la ejecución de una iteración implícitamente están definidas dos modos de funcionamiento; el modo reactivo para el presente y el modo planificado que genera un plan para un breve e inmediato futuro. El propósito del plan es principalmente brindar a los sistemas de la arquitectura una pista para asignar los recursos de una manera más eficiente que en la forma reactiva por medio de considerar los cambios de configuración del hardware que pudieran venir.
La base de la planificación son las instantáneas descriptivas del futuro las cuales son generadas por los recorridos derivados de las evoluciones descriptivas correspondientes a cada entidad. Las evoluciones prescriptivas para generar las instantáneas planificadas simplemente toman los datos de las instantáneas descriptivas del futuro y generan una instantánea prescriptiva tal como en el caso presente, de lo cual se desprende que no hay cambio en el algoritmo de generación.

Subloque *perform*

El propósito del subloque *perform* es determinar el comportamiento de las entidades para las cuales se generan instantáneas. El propósito de uso -i.e. descriptivo,

prescriptivo o especulativo- que tomarán estas instantáneas será el mismo que el que tenga definido la evolución.
El sub bloque perform tiene la siguiente forma:

```
perform on <lista de entidades>
      do
            ...
end
```

Todas las entidades de la instrucción *perform* deben pertenecer al contexto. Únicamente sobre las entidades declaradas en la clausula *on* está permito asignar valores a sus dimensiones o atributos. La lectura de dimensiones está permitido para todas las entidades. Además los atributos de las instantáneas tienen un control de acceso de modo que los atributos *set* sólo pueden escribirse y los *get* sólo leerse -ver el apéndice "Estructura de las instantáneas"-.

Subloque *switch milestone*

El propósito de *switch milestone* es aislar los hitos que determinan los cambios en la ejecución del recorrido. Está formado por una secuencia de condiciones sobre las instantáneas descriptivas de las entidades de la sección *context* y las variables de la sección *state* del recorrido, y tienen asociado un código que se ejecutará cuando la condición sea verdadera.
En este código se puede cambiar de estilo con la instrucción *changestyle*, leer las instantáneas descriptivas y modificar las variables del estado del recorrido.
Las condiciones son evaluadas al principio de cada ciclo del bloque *while* siguiendo el orden de escritura. Se disparan todas las que cumplan con la condición y el estado modificado por una afecta a la evaluación de las siguientes. Sin embargo, puede detenerse la evaluación de las condiciones siguientes si se ejecuta una instrucción *break*. Es aquí es también en dónde se establecen los puntos de interrupción con la instrucción setInterrupt sobre entidades declaradas en la cláusula *on* del el subloque perform.

Sub Bloque *rhythm*

Rhythm se utiliza para cambiar de estilo regularmente para generar momentos de alta y de baja demanda de los recursos con el fin que los recorridos subordinados puedan utilizar los recursos cedidos. Contiene una secuencia circular de estilos asociados con un múltiplo de la velocidad de reacción del recorrido. Cuando el estilo sea uno de los que está en la secuencia de *rhythm* entonces ese estilo perdurará vigente por el tiempo indicado en esta sección. Tras el vencimiento se cambia al estilo que le siga en la secuencia por el período asociado al mismo y así sucesivamente recorriendo circularmente la secuencia. Aunque es circular tiene un inicio y los estilos pueden aparecer varias veces en esta. Por eso cuando por un cambio de estilo explícito se entra en algunos de los estilos presentes en la sección *rhythm* la misma se activa en la primera ocurrencia desde el inicio de la secuencia para el estilo activado. Las instrucciones explícitas de cambio de estilo tienen prioridad sobre los cambios propuestos por la bloque *rhythm* y de lo anterior se desprende que, en caso que se pase explícitamente a un estilo no declarado en este bloque, la única forma de volver a activarlo será pasar explícitamente a uno de los estilos de la sección.

Bloque start

Este bloque se ejecuta la primera vez que el recorrido se inicia lo cual se produce cuando el recorrido padre ejecuta el comando *start*. También, se ejecuta cuando se utiliza la operación de *transition lever* -ver sección *syncronization*-, la cual sólo puede ejecutar un comando *start* y en este caso la activación puede venir con una restricción de ejecución para la próxima instantánea a generar.
En este bloque se puede acceder al estado del recorrido y las los atributos temporales *time* y *processtime* corresponde a los que tenga definido el comando *start*. Dichos

atributos se usarán para determinar las instantáneas descriptivas a leer y su objetivo principal es configurar el estado privado del recorrido. Para llevar a cabo su trabajo se puede servir de variables automáticas de bloque que se destruirán tras el fin de la ejecución del mismo.

Bloque finish

Se ejecuta después de que se recibe un comando de *stop* y antes de reiniciar el ciclo del bloque *while*. El propósito de este bloque es básicamente modificar las entidades abstractas que mantienen el estado compartido de todos los recorridos. Sólo está permitido por lo tanto modificar estas entidades ya que el estado del recorrido será inmediatamente destruido; sí, en cambio, se puede leer el estado del recorrido. No se puede realizar *starts* ni *stops* ya que automáticamente se enviará un *stop* a todos los recorridos subordinados. El *time* del bloque depende del tiempo del comando *stop*. Si está en el futuro no se destruye la instancia pero se considera en las instantáneas planificadas y además se envían *stops* a los subordinados con el mismo tiempo que el recibido en el *stop* propio. Recién cuando se alcance el presente el recorrido se destruye. Están permitidas las variables automáticas de bloque.

Bloque resource

Objeto ResourceError

El objeto ResourceError es un objeto de sólo lectura que completa automáticamente el lenguaje y que contiene los problemas con la materialización de las instantáneas. Básicamente son los requerimientos que no estén siendo cumplidos. Este objeto tiene asociado un iterador que permite recorrerlo ya que está estructurado como un árbol. El nodo raíz es siempre el recorrido donde está siendo usado. Sin embargo, si el problema sucedió en un recorrido subordinado y el mismo ejecuta una instrucción raise entonces el objeto ResourceError del recorrido superior tendrá el *path* de activación hasta llegar al recorrido con el inconveniente.

Una vez en el recorrido de origen el árbol del objeto ResourceError continua, a través de los estilos y rasgos, hasta llegar a los requerimientos que son siempre las hojas. El objeto ResourceError sólo puede ser utilizado en el bloque resource.

Detalle del bloque Resource

Este bloque se relaciona con la falta de recursos para materializar las instantáneas. El problema de materialización puede ser tanto sobre las instantáneas necesarias como con las generadas.

Antes de ejecutar todo ciclo implícito de *while* se evalúa si se satisfacen con los requerimientos del estilo vigente o si existe algún mensaje en la cola de errores de materialización. Si tras la evaluación se encuentra algún problema entonces se actualiza el objeto ResourceError y se activa el código del bloque resource.

En el caso de detectar problemas con la instantánea generada entonces todas las instantáneas posteriores generadas que dependan de la que falló son eliminadas ya que se supone que su sustento ya no es válido y se reinicia el ciclo de generación.

El tratamiento del problema se basa en explorar el objeto ResourceError y en función de eso actuar de alguna de las siguientes maneras:

- *Raise*, es la instrucción default si se llega al final del evento sin haber tomado una acción o, si está ausente la sección, pasa hacia “arriba” el problema provocando que se dispare este mismo evento pero en el recorrido llamador. En caso de llegar al recorrido *Main* y que ahí tampoco sea procesada entonces se descarta y se llama al reset del robot. Tras el *raise* el recorrido se bloquea esperando un *continue* o un *stop*. El recorrido superior puede:
  - cambiar de estilo para actuar sobre el subordinado a través del flujo y enviar un *continue.*

- hacer un *continue* directamente que implica un retry -ver a continuación- pero desde el recorrido llamador lo cual puede ser peligroso ya que podría quedar en un ciclo infinito.
- decidir terminar con el recorrido problemático con un *stop*.

- *Retry*, es la instrucción que reintenta con el estilo activo. Aquí antes de ejecutarla puede realizarse un *changestyle* o utilizar variables de estado predefinidas asociadas a *retry* que cuenten los reintentos y la duración de la falla.

Sección syncronization

Introducción

Lo más esperable es que el comportamiento global del robot dependa de varios recorridos para su concreción. De ahí que resulte necesario coordinar y sincronizar la generación de las instantáneas que producen esos recorridos [9][11]. Así, si la aplicación del robot ha sido organizada teniendo en cuenta varias evoluciones, una posibilidad es que alguno de los recorridos sea el que deba marcar pautas de actuación y otro, el que las ejecute. Dicha posibilidad es la que se seleccionó para esta arquitectura.

La columna vertebral del mecanismo de sincronización es utilizar una sincronización asincrónica con el propósito de que las velocidades de ejecución de los recorridos se adapten a las circunstancias del momento. Por lo tanto, para llevar adelante la tarea de una forma coordinada es indispensable incorporar al código de las evoluciones ciertos *puntos de control* que evalúen la marcha del recorrido e informen a otro recorrido -al que controla- sobre la ejecución del primero. Como ya se mencionó toda evolución tiene definida una velocidad adecuada para la tarea que desempeña y es por eso conveniente que el método elegido para la sincronización de los recorridos no las obligue a modificar dicha velocidad sólo para adaptarse a la velocidad del recorrido controlador.

La forma de sincronización asincrónica establece que el orden de ejecución de las distintas acciones no dependen de un reloj sino de condiciones declaradas explícitamente que tienen que hacerse verdaderas para que las acciones puedan ejecutarse. Y, puesto que el objeto central de trabajo son las instantáneas, es ahí donde hay se aplica la forma de sincronización. El lenguaje hace casi transparente la implementación de la sincronización que lleva a cabo la arquitectura; la misma consiste en asignar a cada instantánea un identificador e incorporar a cada una de ellas un conjunto -posiblemente vacío- de identificadores de otras instantáneas que deben llevarse a cabo antes de la instantánea en cuestión. El caso más simple es para las instantáneas de un mismo recorrido; estas se acomodan como un tren indicando como predecesora justamente a la generada previamente.

Ya se sabe que los recorridos generan instantáneas y que los recorridos están organizados en una estructura de árbol que surge de la activación que hacen los recorridos padres de sus hijos, de lo cual se deduce, que *Main* es la raíz del árbol. Además las instantáneas generadas por un mismo recorrido tienen una secuencialidad innata con la previamente generada; así, si se consideran varias instantáneas por recorrido y se visualiza a cada una como perteneciente a un mismo árbol con aquellas otras que están aproximadamente en el mismo período temporal se verá que se conforma como una sucesión de árboles como si fuera cada árbol una rebanada de un pan.

Es claro que en este andamiaje existen tantos hilos de ejecución como recorridos activos y que todos son independientes y es precisamente la sincronización lo que provoca la conformación de dependencias entre las instantáneas para que de esa forma emerja un comportamiento total coherente.

La forma de sincronizar consiste en que algunas instantáneas deban esperar a otras estableciéndose de ese modo dependencias entre instantáneas tanto de un mismo árbol como con otros.
La sincronización de los recorridos presupone que las entidades son globales. La misma se lleva a cabo pasando una condición espacio temporal sobre las entidades a todos los recorridos subordinados -hijos, nietos, etc.-. Los subordinados cada vez que generan una instantánea verifican la condiciones recibidas y si alguna se hace verdad pasan el control al recorrido que tiene en su código escrita la condición. El recorrido superior recibe el control de sus subordinados y decide como organizar la precedencia de las instantáneas de los subordinados que se generaron cuando las condiciones suministradas se hicieron verdaderas. Así se sincronizan las instantáneas generadas en distintos recorridos.
En esta sección se describen los elementos del lenguaje que posibilitan describir la sincronización necesaria de los recorridos de una aplicación.

Idea

El método de sincronización propuesto en esta arquitectura está inspirado en las redes de Petri. Las redes de Petri son una herramienta para describir el comportamiento asincrónico de sistemas que se conforman de una red de *lugares* y *transiciones* -como nodos- y arcos dirigidos que los vinculan. Además en los *lugares* pueden haber *tokens*. Las transiciones se disparan cuando en todos los *lugares* que la preceden existe al menos un *token* -que será removido de ese *lugar* tras la activación- mientras se coloca un *token* en cada *lugar* que sea consecuente de la *transición*. No está permitido vincular directamente una transición con otra transición o un lugar con otro lugar.
Tres extensiones se realizan sobre las redes de Petri. La primera es que los *tokens* -que en la reelaboración se llaman *packages*- llevan un contenido a diferencia de los tokens que son indistinguibles entre ellos, la segunda es que los *package* para ser liberados deben satisfacer una condición -definida en el elemento del lenguaje *trigger*- y la tercera es que las transiciones para dispararse además de que haya en cada uno de sus arcos adyacentes un *package* los mismos deben intersectarse temporalmente en la dimensión *time* considerando sus deltas.

Descripción de la sincronización

Es claro que la sincronización se necesita llevar a cabo cuando se decide utilizar un recorrido subordinado y controlar su actuación y el modo de llevarlo a cabo es por medio de *triggers* y de *transitions*. En los *triggers* se definen los “puntos de control” citados al comienzo. Estos puntos de control son expresiones booleanas sobre las instantáneas descriptivas de las entidades declaradas en la evolución dónde se escribe el *trigger*. Tras la activación de un *trigger* se envía un *package* a una *transition.* En sí las *transitions* son los lugares donde funcionalmente los *packages* de los recorridos esperan para sincronizarse. A su vez tras la activación de una *transition* un nuevo *package* se genera en función de los *packages* que recibió la misma y estos últimos son destruidos.
A continuación se muestra un esbozo de la sección *syncronization* de una evolución que hace uso de dos recorridos subordinados antes de proseguir más profundamente sobre los elementos del lenguaje que se encargan de la sincronización.

```
syncronization
   about RecorridoABC
      trigger sesgado_izquierda
         ...
      end
      trigger sesgado_derecha
         ...
      end
      ...
   end
   about RecorridoQWE
      trigger demasiadocerca
        ...
     end
     ...
   end
   transition iniciarRecuperacion
      ...
   end
end
```

Cuando en una evolución se utilizan recorridos subordinados y es necesario sincronizar su actuación entonces se crea un bloque *about* por cada recorrido que deba sincronizarse y dentro de este bloque se escriben los triggers para el recorrido en cuestión. El bloque *about* se nombra de igual que el recorrido que representa, de hecho es llevada a cabo una verificación estática en el código para comprobar que los nombres de los bloques *about* se correspondan con los nombres de los recorridos declarados en la sección *context* de la evolución. La excepción a esta regla se aplica en el bloque *about* llamado *this* -palabra reservada del lenguaje- que agrupa los *triggers* para del recorrido propio de la evolución. Los *trigger* tienen un nombre que los identifican y el mismo es utilizado en las *transitions* para representar a los *packages* que *triggers* envían.

La verificación si debe dispararse un *trigger* se lleva a cabo en el hilo de ejecución del recorrido llamado igual al del bloque *about* donde el *trigger* este escrito; así como en todos los hilos de ejecución de los recorridos que son descendientes de este. Esto es así para que cuando un recorrido se implemente utilizado a su vez otros recorridos las expresiones para el primero no dejen de evaluarse mientras los recorridos descendientes hacen su cometido. De esta forma las expresiones de los ancestros siempre son tenidas en cuenta. Además la evaluación en los recorridos subordinados permite mantener separadas las velocidades de los mismos de sus ancestros de forma que el recorrido que evalúa la expresión y el que la utiliza cuando se haya hecho verdadera puedan marchar cada uno a su propia velocidad. Si en cambio la expresión fuera evaluada por el recorrido que la necesita entonces este último debería marchar a la misma velocidad que el recorrido subordinado para poder hacer frente a cada una de las instantáneas que genere el subordinado.

Fases de los recorridos

Los recorridos durante su funcionamiento pasan por las siguientes fases siempre y cuando haya elementos que procesar:

1. se genera una instantánea
2. se construye el *package* para la instantánea
3. se verifica, por el orden de escritura en el código, los *triggers* del bloque *about this* si se disparan
    a. si sí se dispara alguno, se envia el *package* a la *transition* y no se generan nuevas instantáneas hasta que se active la *transition*
4. se verifica si se activa alguna *transition* definida en la evolución del recorrido

    a. si sí se activa una *transition*, construir el *package* fruto de la ejecución del código de la *transition* -tiene en cuenta si hay *package* por la instantánea del recorrido-
5. se verificar si alguno de los *triggers* de los ancestros se dispara. El sentido de evaluación es desde el padre hasta *Main*.
    a. si sí se dispara alguno, enviar el *package* generado recién en la *transition* y bloquearse hasta recibir la respuesta de la *transition* del ancestro.
6. si hay algún *package* -es decir no fue enviado a un ancestro- enviarlo al SIET.

El propósito de este orden se basa en que la especificación del comportamiento global del robot se inicia en los recorridos de las hojas del árbol de activación de *Main.* Así la especificación del comportamiento se va especializando a medida que sube hacia la raíz. Sin embargo, en cada nivel se verifica, por medio de las condiciones de los *triggers* de los ancestros, el desarrollo mismo. Por lo tanto, estas condiciones limitan el comportamiento de los descendientes y determinan así si toda la especificación del comportamiento construido hasta ese nivel es enviada a un ancestro para su procesamiento o si ya está lista para enviarla al SIET .

Los *triggers*

Los *triggers* se conforman de tres cláusulas que definen el significado del mismo: *situation*, *when* y *go*. La primera cláusula es para denotar la expresión booleana construida con instantáneas descriptivas que verifica si el robot está o no en una situación dada. La cláusula *when* califica la expresión de *situation* cuando se hace verdadera y los tipos posibles de calificación son *first-time*, *each-time* y *last-time*. El resultado de la calificación del *when* será lo que provoque en última instancia que se dispare o no el *trigger*. Así cuando la situación es verdadera se habilita la evaluación del *when* y *first-time* será verdadera la primera vez que ocurra o cuando se haga verdadera la situación después de haber sido previamente evaluada como falsa; *each-time* se hace verdadera cada vez que la situación sea verdadera y *last-time* se hace verdadera la primera vez que la situación es falsa después de haber sido verdadera.

```
syncronization
   about RecorridoABC
       trigger sesgado_izquierda
                        ...
             end
       trigger sesgado_derecha
                        ...
             end
             ...
   end
   about RecorridoQWE
             trigger demasiadocerca
                        situation around Self
                           (distance Self Pelota <
                       Pelota.diametre / 5)
                        when first-time
                        go iniciarRecuperacion
                   end
             ...
   end
   transition iniciarRecuperacion
      ...
   end
end
```

El cuadro de abajo muestra en 12 instantes consecutivos cómo es la calificación que hace *when* según el valor de verdad de la expresión de *situation*. Empezando con tres instantes donde la expresión es falsa.

instante	1	2	3	4	5	6	7	8	9	10	11	12
condition	○	○	○	☼	☼	☼	○	○	○	☼	○	○
first time	○	○	○	☼	○	○	○	○	○	☼	○	○
each time	○	○	○	☼	☼	☼	○	○	○	☼	○	○
last time	○	○	○	○	○	○	☼	○	○	○	☼	○

Así, dónde aparece un “sol” en la tabla indica el disparo el *trigger* que envía el *package* que este procesando en el recorrido a la *transition* indicada en la cláusula *go*. En ese momento el recorrido se bloquea a esperar la resolución de la *transition*.
Los resultados históricos de la *situation* de cada *trigger* es mantenido automáticamente por el motor del lenguaje para que cuando se evalúen las condiciones de la *situation* en los recorridos subordinados estos puedan determinar si corresponde o no disparar el *trigger*. Así el *when* no se evalúa sólo con los resultados locales sino que considera toda la rama de recorridos descendientes del recorrido que tiene escrito en su código el *trigger*.

Expresión de *situation*

Las expresiones de *situation* tienen que tener la capacidad de ser evaluadas en cualquier recorrido que sea descendiente del recorrido de la evolución en cuyo código está escrita la *situation*. El modo en como se propone para llevarlo a cabo es declarando en cada *situation* una entidad de referencia. Así el algoritmo que vincula las instantáneas de la expresión podrá evaluarse en cualquier recorridos.
Las *situations* no realizan pedidos de entidades sino que son dependientes de las instantáneas descriptivas que estén disponibles en el modelo. Por lo cual, los recorridos deberán hacer los pedidos si bien, es de esperar, que las entidades de las *situation* estén involucradas ya en los recorridos para cumplimentar sus propios fines.
En caso que la *situation* no puede ser evaluada porque no están disponibles las instantáneas el resultado será *null* y se disparará el manejo de error.

Construcción de los *packages*

Como ya se introdujo el paquete es el análogo al *token* de las redes de Petri y se diferencia de este en que el paquete tiene un contenido. Así tiene un identificador, un grafo de instantáneas ordenadas por la relación de sincronización, comandos a aplicar y datos para que pueda ser relacionado con paquetes previos y/o posteriores. Los paquetes se construyen en base a las siguientes reglas:

- toda instantánea generada es un paquete.
- la composición de las operaciones de las transiciones reciben paquetes y generan uno nuevo fruto de la composición.

El fin de las transiciones es sincronizar instantáneas mediante la manipulación de las restricciones de ejecución, la eliminación de instantáneas, la detención de recorridos así como su arranque. A diferencia de las condiciones de los *triggers* las transiciones se ejecutan sólo en el hilo de ejecución del recorrido en donde está escrita. Las instantáneas cubiertas por una transición son todas aquellas generadas por el mismo recorrido en la cual se ejecuta más la totalidad de las generadas por los recorridos descendientes de aquel.
Desde la vista en el lenguaje una *transition* es una bloque de código que se relaciona con los *triggers* de la misma sección *syncronization* en donde está escrita por medio de la clausula *over*. Los *over* indican a las *transitions* de qué *triggers* esperan recibir los *packages.*

```
syncronization
  about RecorridoABC
     trigger sesgadoIzquierda
                              ...
     end
     trigger sesgadoDerecha
        ...
     end
     ...
   end
   about RecorridoQWE
     trigger demasiadoCerca situation around Self
                          (distance Self Pelota <
                              Pelota.diametre / 5)
                              when first-time
                              go iniciarRecuperacion
     end
     ...
   end
   transition iniciarRecuperacion
      over(RecorridoQWE.demasiadocerca,
            RecorridoABC.sesgadoIzquierda)
         ...
   end
end
```

Tras el *over* sólo puede haber una instrucción que será una composición de las operaciones elementales de sincronización aceptadas aquí. Los *packages* son representados en las operaciones por los nombres de los *triggers* que figuran en la cláusula *over* y una verificación estática es realizada por el lenguaje para verificar que estén definidos como recorridos y *triggers*.
En la escritura de la sentencia de composición se verifica que en dicha expresión se de una y sólo una vez la ocurrencia de cada *package* definido en el *over*. De esa forma se evita que la red de dependencia entre instantáneas sea circular lo cual carece de todo significado físico señalando bloqueos perpetuos.
Las operaciones que se pueden utilizar en la instrucción de composición en la *transitions* son:

- Arrange (*package* A, *package* B): Establece una sincronización entre los paquetes A y B. Así construye un *package* vinculando las instantáneas supremas del paquete A como predecesoras de las instantáneas ínfimas del paquete B.
- Inhibit (*package* A): Elimina el efecto del paquete A pero no elimina lo que generó al paquete. Así construye un *package* sin instantáneas y con un comando *delete* para cada instantánea ínfima de cada recorrido que aparezca en el *package* A.
- Conclude (*package* A): Promueve el efecto del paquete A pero detiene lo que lo generó. Así construye un *package* con las instantáneas del paquete A y con un comando *stop* para cada recorrido que aparezca en el *package* A. Un *stop* a un recorrido inexistente no provoca un error dado que ya se supone que fue destruido. Los valores temporales para el *stop* se toman de la primera instantánea del *package* para el recorrido en cuestión si esta existe o sino de la predecesora al *package*. En caso que no haya un *package* previo ocurre un error.
- Abort (*package* A): Es una mezcla entre *inhibit* y *conclude*, elimina el efecto del paquete A cómo así también detiene lo que lo generó. Así construye un *package* sin instantáneas, con un comando *delete* para cada instantánea ínfima de cada recorrido que aparezca en el *package* A y con un comando *stop* para

cada recorrido que aparezca en el *package* A. El comportamiento asociado al comando *stop* es idéntico al informado para el *conclude*.

- Lever ({llamado de recorrido}): Integra un nuevo recorrido a la sincronización. Así construye un *package* sin instantáneas y con un comando *start* sobre el recorrido del argumento arranca el recorrido. Sólo en el papel de ínfimo puede ser usado el resultado de *lever* sino provoca un error de composición ya que no tiene elementos supremos conocidos -las instantáneas serán las generadas por el recorrido del argumento cuando este se inicie-. En el caso que el recorrido ya esté activo se provoca un error.

Por cada entrada declarada en el *over* de una *transition* se establece una cola para guardar temporalmente los *packages* que arriben. La *transition* se activa cuando los primeros *packages* de todas colas se intersecten la dimensión *time* considerando su delta. En las colas de espera de las *transition* también se verifica el *processtime* de los *packages* y aquellos que no tiene más sentido porque se han vencido se eliminan y se avisa al recorrido que lo envió de este hecho por medio de un error en el bloque *resource*.

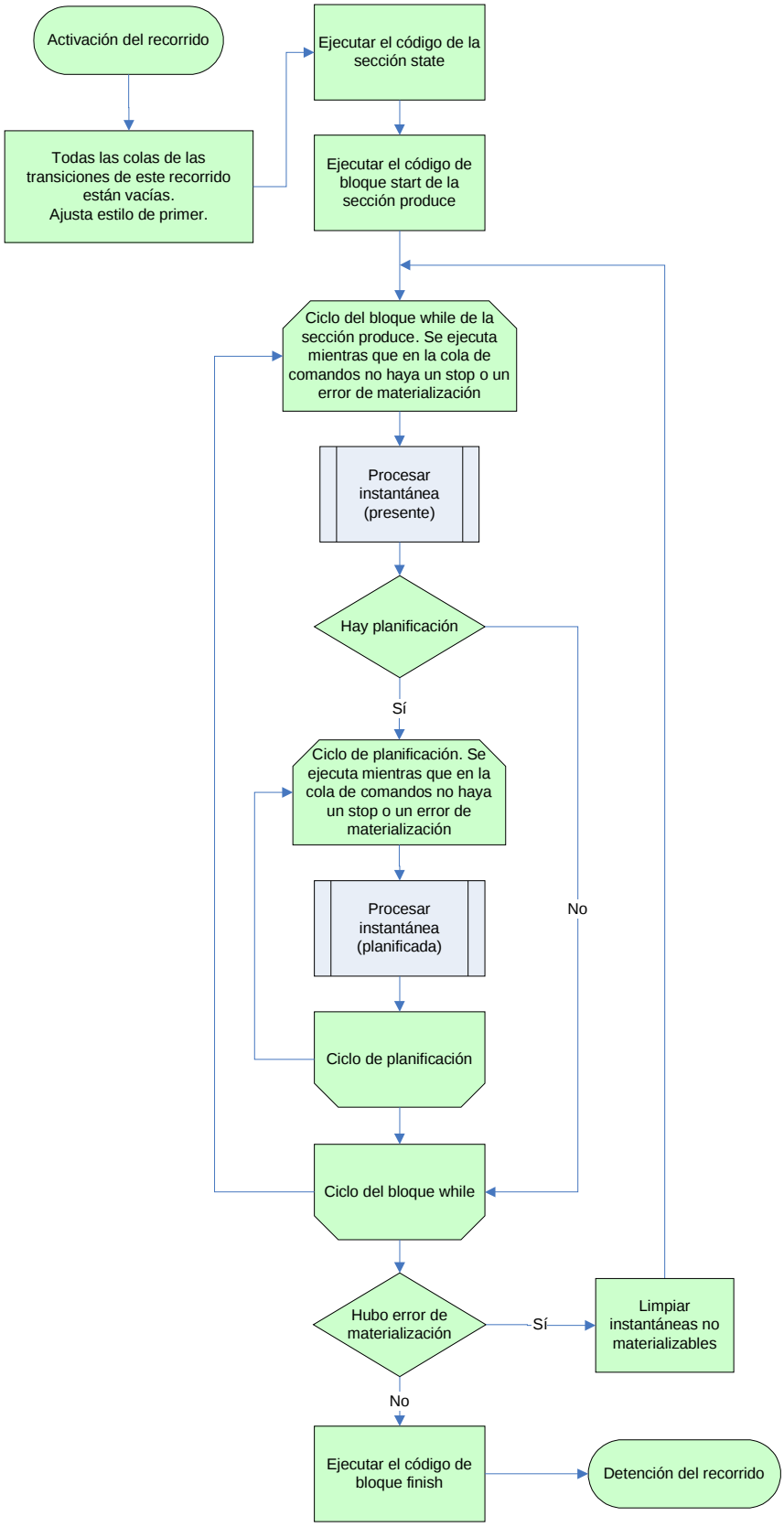
Una vez procesadas las primeras puede hacerse lo propio con los siguientes *packages* que esperarían detrás de las primeras procesándose de ese modo las instantáneas del futuro. Asimismo la dimensión *time* del *package* generado por la *transition* será el período del delta en que se intersecten todos los *packages* que la activaron.

Finalmente, para concluir con la construcción del package si en el recorrido hay un packages por la generación de una instantánea propia y otro por la composición de los descendientes y el de la instantánea propia no está en la composición entonces se agrega al packages de la composición con un hilo independiente.

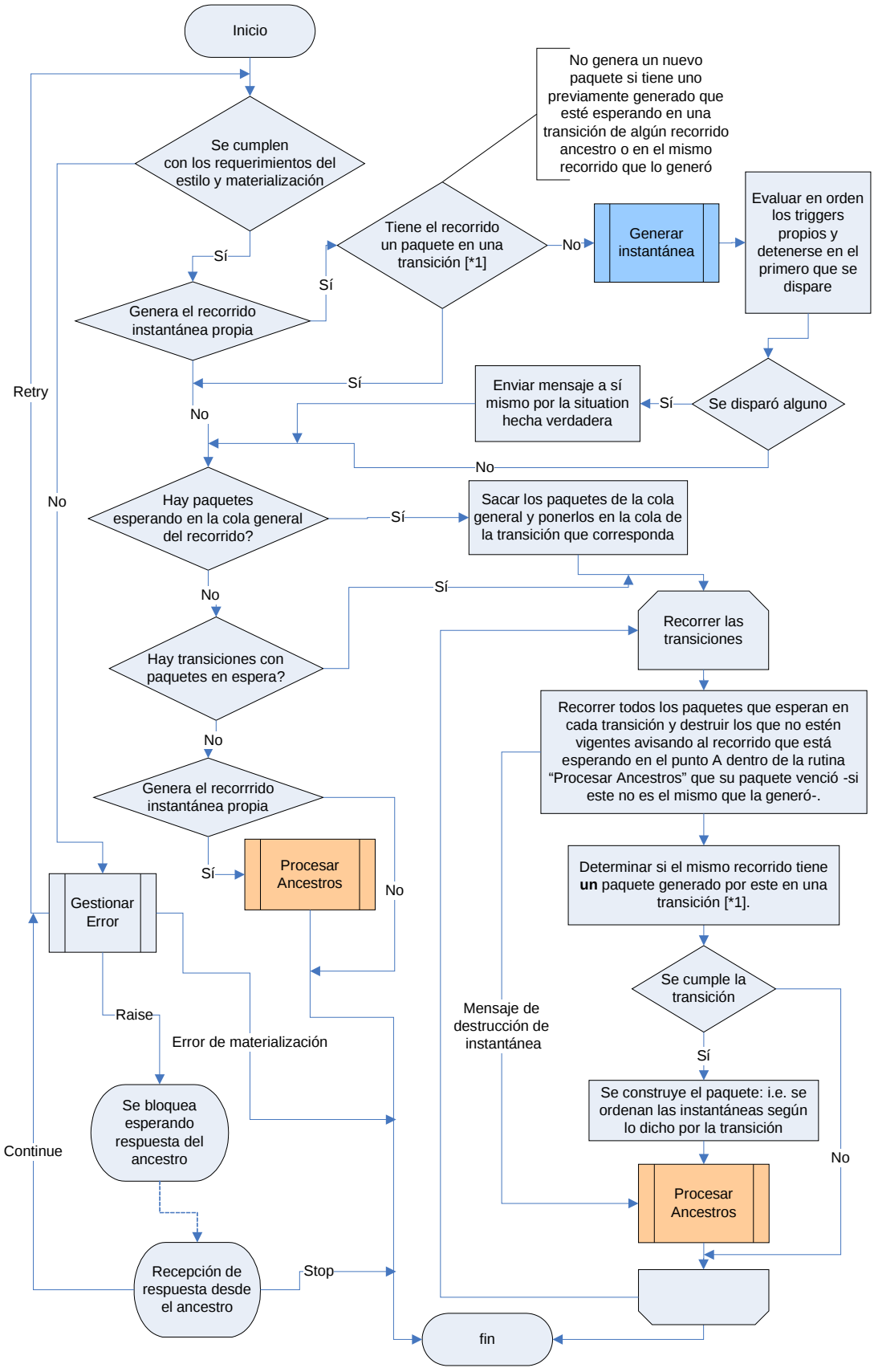
Implementación de los triggers y transitions

Con el fin que la evaluación de las condiciones de los *triggers* sean evaluadas en forma eficiente se mantiene una estructura de datos que remeda el árbol de activación y en dónde en cada nodo está el resultado de la evaluación de los situation de cada recorrido. De forma que para determinar cual es contexto global de sincronización cada recorrido debe navegar por un árbol tantos nodos como la cantidad de ancestros tenga hasta *Main*.

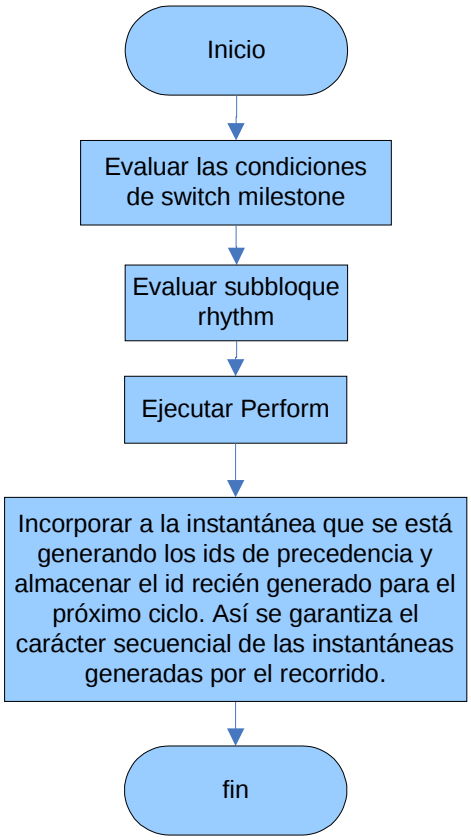
A continuación se muestran cuatro diagramas de flujos que representan el comportamiento de sincronización de los recorridos. Se parte de uno global con el comportamiento del flujo de control de un recorrido; luego dentro de este diagrama se llama a una subrutina que contiene el diagrama del flujo de control del proceso de una instantánea. Finalmente este último diagrama se refina con dos diagramas de flujo más que muestran la generación de una instantánea y la sincronización entre recorridos de distintos niveles del árbol de activación.



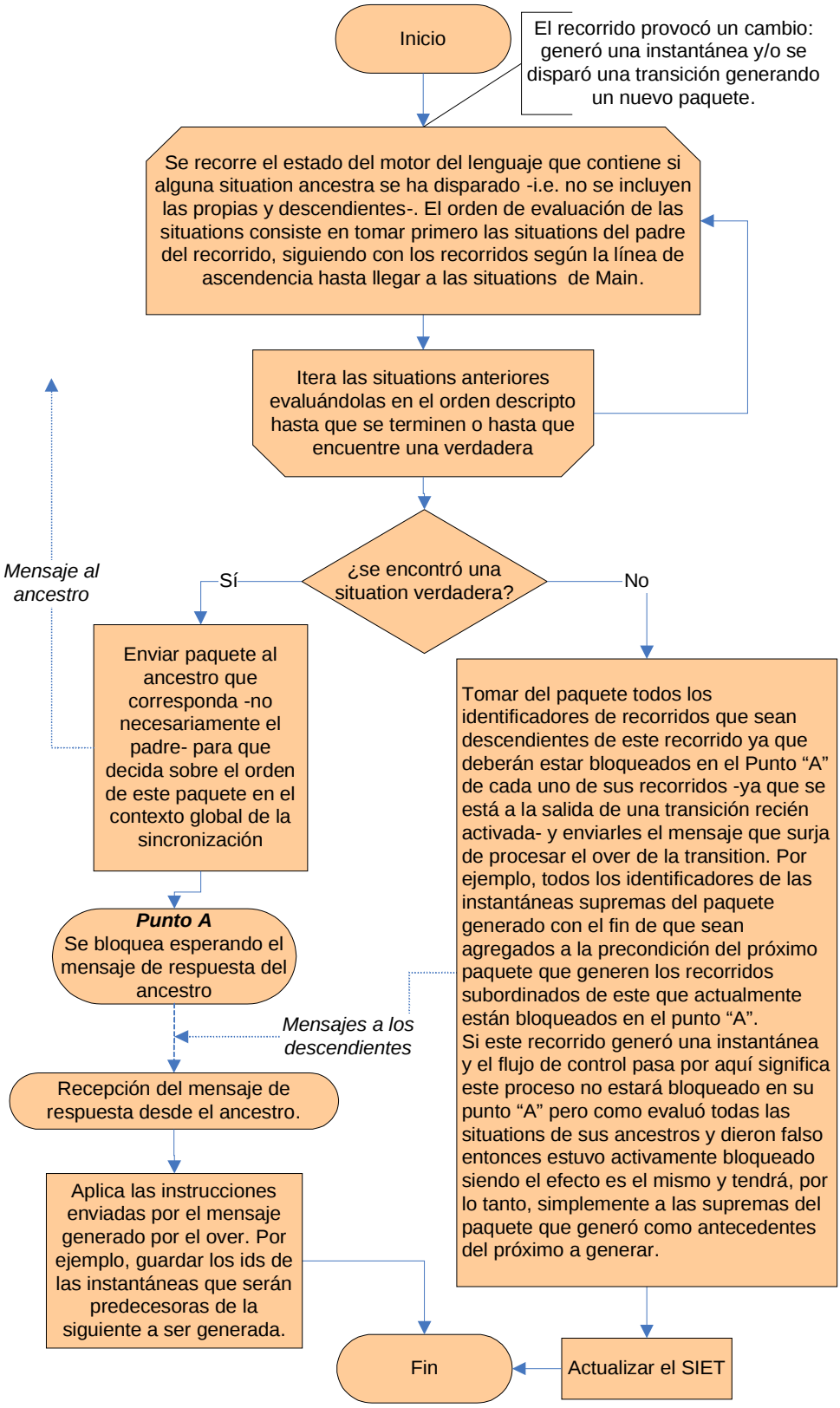
Flujo de control de un recorrido



Procesar una instantánea



Generar Instantánea



Procesar Ancestros

Sistema de percepción (SP)

Objetivo

El objetivo del sistema de percepción es reconocer las entidades y determinar la posición en el espacio de las mismas.

Introducción al SP

En su funcionamiento el sistema de percepción interpreta los datos actuales provenientes de los sensores y también utiliza los ya interpretados, los temporales, los estimados y los supuestos provenientes del sistema de integración espacio temporal (SIET) con el propósito de determinar qué entidad se reconoce, en qué posición y con qué confiabilidades [2][4][5][6][10].
El SP consta de un proceso central que distribuye el trabajo y de tres tipos de procesos que llevan a cabo el trabajo en sí organizados en un capa por tipo de proceso. En la capa superior habrá un proceso por entidad que se esté procesando en el SP. Es a un proceso de esta capa al cual el proceso de distribución les entregará las solicitudes de instantáneas descriptivas -enviadas por el SIET- para una misma entidad.
Los flujos internos de comunicación del SP establece que la capa superior se comunica dentro del SP sólo con la capa del medio, a su vez la capa del medio internamente sólo con la capa superior y con la inferior y la capa inferior sólo se comunica con la capa del medio. Las tres capas tiene comunicación con otros sistemas en particular con el SIET, el SMO y el SGH.
De este modo, la profundidad de la interpretación de los datos fluye de abajo hacia arriba y las señales de control que guían la interpretación lo hacen en un sentido de arriba hacia abajo. Todas las capas tratan solamente con datos vectoriales y no, con datos crudos del sensado. Es función del sistema de gestión de hardware -SGH- convertir los datos sensados en datos vectoriales.
Todos los componente del SP se comunican con el SIET con un sistema de coordenadas ego céntrico lo que significa que la entidad de referencia para el SP es el propio robot, esto es, la entidad *Self*.

Hipótesis

Una hipótesis es básicamente una solicitud de instantánea descriptiva que se le asocia una posición a dónde ir a verificar si allí se encuentra la entidad. Puede ser común que haya varias hipótesis concurrentes para una misma solicitud de entidad. Esto se da por ejemplo cuando se genera una para cada instantánea especulativa.
La hipótesis compilan los requerimientos de varias solicitudes de modo que una misma hipótesis puede tener importancias distintas para distintas dimensiones e identificación -ver sección del SIET *Brindar información descriptiva y especulativa sobre las entidades* pág. 51-.

Intérprete de contexto (nivel superior)

Objetivo

El objetivo del intérprete de contexto es organizar conceptual y semánticamente el espacio alcanzable por los sensores del robot.

Descripción

El intérprete de contexto es el nivel más abstracto de los tres que maneja el SP como así también es el proceso del nivel superior que cumple la función de la fuente

coadyuvante. Las solicitudes de instantáneas descriptivas recibidas del proceso central de distribución contienen los requerimientos de confiabilidad para la posición e identificación de la entidad así como también la importancia de la fuente que realizó el pedido. Así el intérprete de contexto puede calcular una prioridad para cada solicitud en función de los requerimientos temporales y de la importancia para priorizar la producción de hipótesis. Por último, el intérprete de contexto se encarga de determinar las instantáneas descriptivas a enviar al SIET como resultados.
En síntesis, dos funciones realiza el intérprete de contexto:

- Determinar, construyendo hipótesis, la posiciones del espacio dónde se podría encontrar la entidad solicitada por el SIET.
- Seleccionar entre los resultados de las hipótesis cual se determinará como la instantánea descriptiva a enviar al SIET.

Generación de hipótesis
El algoritmo más simple para generar hipótesis es distribuir hipótesis espacialmente para cubrir el espacio tal como lo haría una exploración sin utilizar ninguna otra información que pueda hacer más breve la búsqueda. Según la complejidad del algoritmo del intérprete de contexto los siguientes criterios podrían ser utilizados conjuntamente o no para la generación y gestión de las hipótesis:

- Plantear distintas hipótesis no conexas espacialmente basándose en el dominio de aplicación a través de las instantáneas especulativas. Las evoluciones que calculan las instantáneas especulativas pueden considerar la información que esté disponible en el momento para otras entidades así, en función de las relaciones del dominio de aplicación representadas en las evoluciones especulativas, mejorar la especulación de dónde podría estar la entidad buscada. Por ejemplo, en un juego de fútbol buscar la pelota cerca del arco propio bajo el supuesto que el atacante esté llevando a cabo su cometido. El ciclo comienza con la solicitud de instantánea especulativa que le envía el SP al SIET para preguntarle por las instantáneas especulativas después de evaluar la solicitud de instantánea descriptiva en el contexto de sensando actual. El criterio para elegir las especulativas son las confiabilidades estimadas para la identificación y para las distintas dimensiones que la evolución especulativa haya determinado. Así esta evolución puede devolver desde varias soluciones hasta ninguna dada las circunstancias.
- La gestión de la posición de las hipótesis podría ser reactiva ante la confiabilidad. Así podría mover la hipótesis, medir la confiabilidad y volver a mover la hipótesis en la dirección que aumenta la confiabilidad.
- Se podría mantener un estado cuando verifica que en una ubicación no se encuentra evidencia para la hipótesis de modo de no volver a visitarla inmediatamente y en cambio desplazar la hipótesis a una ubicación aún no explorada. Para recordar las posiciones utiliza entidades virtuales en el SIET con la dimensión tiempo acotada de modo que esté en el modelo sólo por ese tiempo.
- En función de la información sobre los sensores disponibles puede plantear varias hipótesis para trabajar en paralelo. Esto tiene en cuenta las regiones óptimas de sensado de los distintos sensores. Por ejemplo, podría establecer un hipótesis lejos para que se busque con una cámara y varias hipótesis cercanas alrededor del robot para que se busquen con sensores IR. En este nivel usa la información de los sensores disponibles para plantear hipótesis pero será el inferior el que tendrá que decidir utilizar uno u otro sensor. Entonces, para poder realizar esto debe obtener las entidades virtuales que

expresan el alcance de los sensores activos desde el SIET y plantear las hipótesis dentro de esas áreas.

A medida que se va obteniendo evidencia con el resultado de las hipótesis las mismas pueden volver a formularse, eliminarse o reemplazarse por otras de acuerdo a la última información recibida. Una hipótesis también se puede eliminar si se retira la solicitud de entidad que la sustentaba.

Elección de instantánea descriptiva

La elección de la instantánea descriptiva que suministra el SP como interpretación de cada entidad solicitada es el producto de la selección entre los resultados de hipótesis de la entidad en cuestión. La selección se basa en la aplicación en orden de los siguientes criterios que ponderan la información proveniente del modelo junto con la del sensado. Los mismos son:

1. Que el resultado de la hipótesis tenga un índice de confiabilidad -lo calcula el integrador de hipótesis- al menos medio en la identificación de la entidad y que se superponga con una instantánea especulativa ya que estas últimas representan el conocimiento del dominio de aplicación.
2. Que el resultado de hipótesis tenga un índice de confiabilidad alto en la identificación.
3. Que el resultado de la hipótesis tenga un índice de confiabilidad bajo en la identificación de la entidad y que se superponga con una instantánea especulativa. Por ejemplo, si el adversario tapa la pelota puede ser que un ruido tenga un índice de confiabilidad mayor que la pelota pero este criterio le resta importancia al ruido al no estar correlacionado con una instantánea especulativa.
4. Que el resultado de una hipótesis tenga sólo un índice de confiabilidad bajo.

Una vez seleccionado el resultado de hipótesis que interpretará a la entidad solicitada se tomarán los valores de las características variables sensadas y se registraran en el SIET en la instantánea de entidad descriptiva recién enviada para la entidad en cuestión con el mensaje "*establecer característica*".

En las instantáneas descriptivas sensadas la restricción de ejecución no refleja la necesidad de materialización de las previas -como es el el caso de todas las demás que son generadas en vez de sensadas- sino que su fin es crear la urdimbre para que puedan incorporarse a las instantáneas que son generadas a partir de las sensadas como así también indicar el transcurso del tiempo y registrar de ese modo el pasado.

La restricciones de ejecución se agrega cuando se selecciona el resultado de hipótesis y la predecesora será la instantánea inmediatamente sensada para la entidad en cuestión, siempre y cuando no se haya interrumpido el sensado de la entidad o que sea la primera instantánea sensada.

Prejuicio

Como ya se adelantó en el sistema de modelización de objetos los prejuicios son artefactos del SP cuyo fin es describir la expectativa de lo que se sensará. El prejuicio no abarca solamente la descripción de la entidad que se quiere sensar sino que aprovecha la hipótesis de la posición de la entidad para describirla en su contexto con lo que el "fondo" pueda contribuir. Debe quedar claro que por más que en un prejuicio pueda haber referencias a varios prototipos el mismo sólo es funcional a una hipótesis de una entidad. Un prejuicio no necesariamente describe a toda la entidad de interés sino que casi siempre se centra sólo en un aspecto en particular de la misma. Ese aspecto trata de ser seleccionado de modo que discrimine en forma efectiva a la

entidad con respecto al fondo supuesto y a las demás entidades que podrían aparecer. Los prejuicios trabajan sólo en una característica y para una región del espacio determinada llamada *espacio incógnito*. Todas las posiciones espaciales de los elementos se determinan por una jerarquía relacional. Así, el primer marco es *Self* que determina la referencia del espacio incógnito. El segundo, es la propia entidad virtual del espacio incógnito en cuyo dominio se posicionan los centros de los prototipos de las entidades que se esperan sensar y finalmente el tercer marco de referencia son los prototipos sobre los cuales los elementos de interés se relacionan.
Los prejuicios son utilizados luego para guiar el sensado y como resultado de eso los mismos serán reconocidos o no. En base a eso es que los prejuicios son diseñados para bien discriminar positivamente o bien negativamente a la entidad. Por positivo se refiere a que la entidad que se trata de identificar debe satisfacer al prejuicio en cambio un prejuicio negativo se utiliza para descartar a otras entidades y no desperdiciar recursos en aquellas que pueden ser dirigidos a la entidad buscada. Así la entidad que se busca nunca debe satisfacer a un prejuicio negativo. Por último, los prejuicios heredan de la hipótesis el identificador, la tolerancia temporal y la importancia de la fuente objetivo; asimismo también contiene el orden de preferencia de la hipótesis que se determinó en la generación de la hipótesis.
Así un prejuicio se conforma de:

- identificador de prejuicio
- identificación de fuente coadyuvante
- identificador de hipótesis
- orden de preferencia de la hipótesis
- signo (positivo o negativo)
- fuente propósito
- factor de selectividad: -aplica sólo para los de signo positivo- indica con 1 que si se valida el prejuicio se confirma el mismo y con 0 que no se puede determinar nada.
- importancia de fuente propósito
- tolerancia temporal
- una característica
- una zona virtual con respecto a Self llamada *espacio incógnito*
- un conjunto no vacío de identificadores de prototipos
    - para cada identificador de prototipo
        - una posición expresada con al espacio incógnito.
        - un conjunto no vacío de identificadores de elementos que pertenezcan al prototipo y que dispongan de una característica en común con la definida para el prejuicio

El propósito de la entidad virtual espacio incógnito es la meta para desplazar las entidades virtuales del alcance de los sensores sobre aquella de modo que se pueda verificar con el sensado lo declarado por el prejuicio. Por eso es importante que el espacio incógnito ajuste lo más posible con el espacio utilizado por los prototipos que contiene.

Integrador de hipótesis (nivel medio)

Objetivo
Definir prejuicios que discriminen efectivamente a las hipótesis entre sí.

Descripción
El integrador de hipótesis tiene dos funciones principales:

- Crear prejuicios para una hipótesis -fase descendente-.
- Calcular la confiabilidad de la posición y de la identificación de la hipótesis -fase ascendente-.

Fase descendente
El integrador de prejuicios tiene que obligatoriamente construir al menos un prejuicio por hipótesis que recibe pero lo más esperable es que se generaren más de uno. La base para la construcción de los prejuicios son: el prototipo de la entidad necesitada, los requerimientos en resolución y en confiabilidad para cada una de las dimensiones de la posición y la posición que la hipótesis presupone. También se puede utilizar para priorizar la generación de los prejuicios la tolerancia temporal y la importancia de la fuente del pedido que están reflejadas en la hipótesis que genera el nivel de superior; sin embargo, en el algoritmo que se presenta se omite para que sea más sencillo. De hecho, la construcción de prejuicios para la hipótesis es un proceso que puede ser muy sofisticado pero, como no es el propósito de este trabajo la confección del integrador de prejuicios definitivo, sólo se describe una versión muy simple pero funcional para un ambiente suficientemente estructurado.
Un primer paso que puede tomarse es obtener a partir de la orientación de la hipótesis una "firma" del prototipo en cuestión. Una firma son elementos del prototipo que distinguen al prototipo de los demás prototipos almacenados. Esta firma la brinda el sistema de modelización de objetos -SMO- con el servicio "*firma de prototipo*"**.** El servicio recibe como datos un identificador de prototipo, un criterio de selección positivo o negativo y un punto de orientación para el prototipo y devuelve un conjunto de tuplas conformadas por los atributos *id elemento*, *característica* y *valor de dominio de la característica* que discriminan al prototipo recibido con respecto a todos los prototipos almacenados. Además ofrece la probabilidad de que el prototipo suministrado sea identificado positivamente entre todos los prototipos que se reconocen.
Luego puede ser conveniente analizar las entidades que ya estén identificadas en una espacio cercano a la hipótesis de modo de poder utilizar características comunes con la entidad de la hipótesis para facilitar su identificación. Eso se realiza con una consulta al SIET para que informe las entidades que estén presentes en la región solicitada. Obtenida esa información puede usarse el mensaje "*Distancia en característica*" que ofrece el SMO que tomando como argumentos (id característica, valor de dominio, valor de dominio) devuelve un real [0,1]. Un valor de 1 indica la mayor distancia posible entre los dos valores de dominios suministrados y por lo tanto brinda un excelente "contraste" -podría ser por ejemplo un tono grave contra otro muy agudo- y por el otro extremo un valor de 0 señala uno mismo valor de dominio y por lo tanto son indiferenciables.
El SMO ofrece el mensaje "*firma de prototipo para característica*" que a partir de un identificador de prototipo, un criterio de selección positivo o negativo, una orientación y una característica devuelve un conjunto de tuplas de id elemento, y valor de dominio que son muy selectivos del prototipo pasado con respecto a todos los prototipos almacenados y la probabilidad de acierto de ese hecho.
No es necesario que todos los requerimientos sobre resoluciones y confiabilidades para las distintas dimensiones sean satisfechos por una misma característica. Por el contrario, la idea es poder aprovechar distintos sensores disgregando en múltiples prejuicios de diferentes características la validación de la hipótesis. Por ejemplo, si la hipótesis plantea una alta resolución en distancia y en reconocimiento de la entidad, entonces se puede utilizar un prejuicio para identificarla por la característica *color* con

una cámara y un prejuicio con la característica[16] usada por un telémetro para la distancia.
Como el principio de funcionamiento es la comparación de la hipótesis en su contexto espacial incluyendo su "*fondo*" es necesario obtener del SIET las entidades que lo componente. Para eso se utilizan los mensajes "*campo solicitud*" y "*campo respuesta*" el primero envía una entidad virtual para demarcar la zona de interés al SIET y el segundo responde con un conjunto de tuplas conformadas con los atributos (identificador de entidad, id prototipo, posición con respecto a Self) que representan a las entidades que se encontraban en la zona demarcada.

- Se recupera del SMO todas las características que sean alcanzables desde la orientación que tiene el prototipo en la hipótesis.
- Se obtienen del SIET el grupo de entidades que estén cerca y en el "fondo" de la hipótesis.

if la hipótesis no es para una entidad actualmente reconocida **then**
Con la orientación del prototipo de la hipótesis y las características del punto anterior se obtiene un conjunto de firmas negativas con un muy alto nivel de discriminación.
if existen firmas negativas con alto nivel de discriminación **then**
Se ordenan las firmas negativas por probabilidad de acierto para seleccionar las características que más discriminan la situación a fin de descartar hipótesis.
if se encontraron entidades en el contexto del "fondo" **then**
Se buscan en los prototipos de las entidades obtenidas del SIET aquellas que tienen la misma característica que las características que más discriminan a la hipótesis.
if se encuentran los prototipos que satisfacen la condición **then**
Se utiliza el servicio *distancia en característica* del SMO que determina el "contraste" entre los valores del dominio de una característica.
En función de la información colectada se construyen prejuicios y se envían al dispatcher de prejuicios.
else
Como el contexto no comparte características con el prototipo de la hipótesis entonces se construyen los prejuicios sólo con la información de las firmas negativas y se envían al dispatcher de prejuicios.
end if
else
Dado que no hay entidades ya reconocidas en el "fondo" de la hipótesis entonces se construyen los prejuicios sólo con la información de las firmas negativas y se envían al dispatcher de prejuicios.
end if
end if
end if

- Con la orientación del prototipo de la hipótesis y las características halladas al principio se obtiene un conjunto de firmas positivas.
- Se busca en el SMO con el servicio *consulta de zona de sensado* que configuraciones de hardware de tipo sensor utilizan características que estén

[16]La característica del telémetro tendría sólo dos valores posibles: existe o no existe un objeto en la posición.

presentes en las firmas positivas obtenidas y que preferentemente también estén en las entidades del “fondo” recuperadas.

- Se ordenan los requerimientos de la hipótesis por la confiabilidad solicitada en cada una de las dimensiones y en la confiabilidad de la identidad.
- Siguiendo el orden anterior se seleccionan para construir los prejuicios las firmas positivas que tienen una característica que haya un sensor que provea la confiabilidad que necesita el requerimiento y que la característica esté también en el contexto.
- Se construyen tantos prejuicios como sean necesarios para alcanzar las confiabilidades requeridas para el reconocimiento de la entidad solicitada en la hipótesis como así también para la certeza de su posición.

Fase ascendente

El resultado de hipótesis es el producto que compila los resultados de los prejuicios de su hipótesis y es enviado al nivel superior para seleccionar cual hipótesis será promovida a instantánea descriptiva. La compilación de los resultados de los prejuicios utiliza la confiabilidad que cada uno obtuvo desde el SGH y el tipo - positivos o negativos – para calcular la confiabilidad de la hipótesis en los aspectos de identificación de la entidad y de la posición.

La identificación de hipótesis es una de las actividades que cuyo tiempo de elaboración más depende si se está haciendo por primera vez o si se está confirmando una entidad previamente identificada. Sin embargo, se espera que no se realice demasiadas veces la identificación inicial sino que lo previsto es que se confirme una y otra vez la misma.

Un prejuicio tras evaluarse puede venir con un error. En ese caso este nivel puede insistir con el prejuicio o eliminar el prejuicio para que el error no vuelva a ocurrir. Esta situación puede pasar cuando en el nivel inferior se reasignan las zonas de sensado a otros prejuicios y el prejuicio en cuestión se queda sin sensor. Un modo sofisticado de mitigar este inconveniente es que para el cálculo de la confiabilidad se considere la historia de los resultados de prejuicios procesados para una hipótesis. Así, si primeramente se dispone de una conjunción de prejuicios para una hipótesis, y luego uno de los prejuicios permanece inalterado mientras el otro desaparece entonces se puede presuponer que la entidad no cambió y que sólo hubo una reasignación de la capacidad de sensado. Por ejemplo, en una aplicación para jugar al fútbol se podría detectar la pelota con prejuicios sobre el color y la radiación infrarroja. Si el color es sensado con una cámara y la misma se asigna a otra función mientras la huella infrarroja permanece sin modificación entonces se puede suponer que la pelota aun continúa en su sitio sin cambios.

Las características variables de un prejuicio son renderizadas en una máscara en el SMO, el SGH obtiene su valor y con el mensaje *valores de variables* los pasa al SP. El SP obtiene las expresiones lógicas del SMO con el mensaje *descripción de prototipo*. Así con las variables y la expresión ya en el SP se puede evaluar la expresión que utiliza las variables para determinar si el prejuicio es reconocido o no finalmente. Además las variables son pasadas al nivel superior por si el resultado del prejuicio forma parte de la hipótesis promovida estén disponibles los valores a persistir en el SIET para la entidad.

Dispatcher de prejuicios (nivel inferior)

Objetivo

Planificar el sensado de los prejuicios.

Descripción
El dispatcher de prejuicios es un componente del sistema de percepción que debe resolver como función más importante el problema de asignar a través del tiempo y en forma eficiente recursos escasos de modo de obtener el máximo beneficio. Como puede verse no es un dominio exclusivo de la arquitectura propuesta y, puesto que el diseño que se realiza tiene como propósito verificar la factibilidad de la arquitectura, es que se tomó para este componente una solución simple para el mencionado problema. Más adelante soluciones sofisticadas podrán ser consideradas.
El dispatcher trabaja sin mantener un estado o memoria sobre su funcionamiento de modo que se comporta como una estación de un línea de montaje. Así el integrador de hipótesis le envía prejuicios, el dispatcher los procesa y los devuelve con un estado de procesamiento que puede ser de error o de procesado. En ese momento el integrador de hipótesis puede volver a enviar el mismo prejuicio -caracterizado por el identificador del prejuicio- modificado o no. El resultado en sí de la validación del prejuicio es enviado directamente al integrador de hipótesis por el sistema de gestión de hardware.
El dispatcher para determinar la asignación de los recursos de prejuicios contempla dos grupos de criterios:

- la preferencia de la hipótesis.
- la importancia de fuente objetivo.
- la tolerancia temporal de la solicitud de la entidad.
- signo del prejuicio.
- el valor discriminante del prejuicio.

- los sensores que estén ya configurados.
- los movimientos de desplazamiento de sensores.
- el cambio de las configuraciones de hardware.

El primer grupo son los criterios que determinan la necesidad del sensado mientras los segundos criterios son los que constituyen el costo del sensado. Además el primero tiene que ver con información inmediatamente disponible y que tras ser procesada permite encargar el segundo grupo de criterios que se basa en información que hay que obtener y que por lo tanto es más costosa. La propuesta entonces es determinar primero el beneficio en base a la necesidad y luego adentrarse a obtener la información para el segundo grupo.
El beneficio se calcula ordenando a los prejuicios por:

- rangos de tolerancia temporal de la solicitud de la entidad para generar grupos de prejuicios de más o menos la misma tolerancia temporal.
- la importancia de fuente objetivo.
- la preferencia de la hipótesis.
- signo del prejuicio (negativo es primero).
- el valor discriminante del prejuicio descendiente (1 primero, 0 al final).

Una vez ordenados los prejuicios se recorren y se procesan de a uno. Al principio los prejuicios pueden ser rechazados por el tiempo que le toma al robot logra cambiar “el punto de vista” de los sensores. El procesamiento de cada prejuicio consiste de:

- verificar la tolerancia temporal cancelando el procesamiento del prejuicio si ya está vencido.
- tomar el espacio incógnito y la característica y calcular la distancia a las entidades virtuales que determinan el alcance de zonas de sensado activas que tienen la capacidad para sensar el prejuicio.

- Si no se encuentra ninguna zona de sensado buscar en el SMO una zona de sensado que tenga la capacidad de sensar el prejuicio que no sea incompatible con las zonas de sensados activas que ya fueron consideradas para los prejuicios anteriores a este en cuestión.
- Si no se encuentra la anterior zona de sensado informar al integrador de hipótesis que el prejuicio en cuestión no será procesado.
- Si sí se encuentra realizar el cambio de configuración de hardware en la entidad sensor correspondiente.

Puesto que la entidad virtual que describe el alcance del sensor puede ser más grande que los espacios incógnitos a procesar es lo que determina que la estrategia de sensado busque procesar la mayor cantidad de espacios incógnitos posibles siguiendo un orden de procesamiento que aproveche el cubrimiento del alcance de los sensores. Así para llevar a cabo la estrategia del movimiento del sensado se utiliza una heurística cuya idea es:

- seguir el orden hallado de los prejuicios hasta encontrar el primer prejuicio que su espacio incógnito no esté cubierto por una zona de sensado capaz de sensarlo.
- asignar al prejuicio en cuestión la zona de sensado activa más cercana que pueda sensarlo.
- generar instantáneas prescriptivas para desplazar la entidad alcance de la zona de sensado seleccionada para que cubra la entidad espacio incógnito del prejuicio en cuestión sin dejar de cubrir con sus respectivas entidades de alcance los espacios incógnitos de los prejuicios anteriores en el orden. Si no se logra la superposición notificar al integrador de hipótesis que el prejuicio no será procesado. Las instantáneas prescriptivas del SP toman las prioridad de ejecución y temporal del prejuicio.

Cuando la superposición entre una entidad de alcance y un espacio incógnito se logra se envía el mensaje al sistema de modelización de objetos (SMO) para que renderice al prejuicio. En este momento se notifica al integrador de hipótesis que el prejuicio fue procesado satisfactoriamente.
El funcionamiento integrado con el sistema de materialización del lenguaje (SML) y el de integración espacio temporal (SIET) permite subsanar algunos inconvenientes de esta sencilla solución. La asistencia del primero se logra con el uso del sub bloque *rhythm* que permite que la exigencia de los requerimientos sobre las entidades bajen y suban de modo de generar espacios para compartir los sensores y, con el segundo, por medio de los recorridos especulativos para cubrir la falta de disponibilidad de sensores. Así los recorridos del SML controlan indirectamente, en función de cuán exigentes sean con sus pedidos, cuánto permiten que el SP comparta el uso de los actuadores y los sensores. De modo que si los pedidos del SML no son tolerantes -y a veces no tienen que ser lo- el SP no podrá compartir los recursos. Por ejemplo, si en una aplicación para jugar el fútbol un recorrido de importancia pide la identificación de la pelota con una altísima confiabilidad durante todo el tiempo puede pasar que el SP no pueda liberar la cámara para buscar el arco. En cambio si baja la tolerancia temporal y/o la confiabilidad de identificación para la pelota permitirá que la entidad arco obtenga sensores para poder ser identificada y posicionada. En el caso que una entidad no pueda ser sensada entonces el SP puede "*pisar la pelota*". Eso se lleva a cabo al activar el recorrido especulativo de la entidad en cuestión a la vez que se reconfiguran el uso de los sensores asociados a aquella. Así, la instantánea especulativa puede generar una instantánea especulativa de buena confiabilidad al cruzar la información del dominio de aplicación con el hecho de que en un instante

previo la entidad estaba fuertemente posicionada e identificada. Por lo tanto, por más que la entidad en cuestión se haya quedado con pocos sensores, la asistencia de instantánea especulativa permite que se siga informando la instantánea descriptiva generada por el SP. Ya que en las instantáneas descriptivas que entrega el SP se informa como metadato cómo se obtuvo la misma hace en los recorridos especulativos puedan controlar la situación. Por lo tanto, de persistir el “pisado de pelota” por mucho tiempo las instantáneas especulativas comenzarán a bajar su confiabilidad lo que provocará que el SP debe revertir a la situación de sensado previa.

Sistema de actuación (SA)

Objetivo

Convertir la declaración del movimiento para las entidades en comandos de acción concreta sobre los actuadores.

Introducción al SA

La función del sistema de actuación es que se lleven a cabo las instantáneas prescriptivas declarativas que recibe desde el SIET [3][5][6]. Las instantáneas prescriptivas declarativas son generadas por el SML y el SP. El SA también genera instantáneas prescriptivas pero las mismas son procedimentales -más adelante se definen- y del mismo modo que las declarativas deben pasar previamente por el SIET para su integración. Una vez integradas el SIET notifica al SA para que genere los comandos a partir de las instantáneas prescriptivas procedimentales integradas y los envíe al sistema de gestión del hardware (SGH).

Las instantáneas prescriptivas declarativas pueden ser tanto para entidades propias como no propias. En ambos casos es tarea del SA determinar que entidades actuador activar para que estas, actuando sobre las entidades no propias -y las propias que no sean actuador-, logre que se muevan tal como lo solicitan las instantáneas prescriptivas declarativas recibidas. Así, el funcionamiento del SA, consiste en convertir las instantáneas prescriptivas declarativas en instantáneas prescriptivas procedimentales, esto es, en casi comandos para el SGH y luego controlar la ejecución de los comandos realizada por el SGH.

Para acotar el tamaño de este trabajo el SA está simplificado y cubre sólo el caso de robots estructura cilíndrica con dirección diferencial de dos ruedas. Tal decisión afecta a la capacidad de movimientos posibles del robot y determina que el único modo de actuar sobre el mundo sea empujado a las demás entidades o ante poniéndose a estas.

Definición de conceptos

Contexto de acción

El contexto de acción contiene a las entidades propias y no propias que se encuentran cercanas del robot y por lo tanto podrían participar en la acción. El mismo se construye alrededor de la entidad Self ya que la acción es inherentemente ego referenciada. Es función del SIET constantemente informar al SA qué entidades no propias entran y salen del contexto de acción ya que las propias, por definición, siempre estarán incluidas.

Cuando una entidad es incluida se recupera desde el SMO la información mecánica de la misma y recíprocamente cuando una entidad sale del contexto de acción la información mecánica es eliminada del contexto. Se registra básicamente los atributos mecánicos de las entidades como son la masa y la posición, la velocidad y aceleración en cada dimensión.

Motor de dinámica

El motor de dinámica es uno de los componentes principales del sistema de actuación. El mismo calcula las próximas posiciones de las entidades del contexto de acción en función de las fuerzas aplicadas en su centro de masa.

Detector de contacto

El detector de contacto es otro de los componente principales del sistema de actuación. Su función es determinar cuando se producen contactos entre entidades y estimar la fuerza de contacto involucrada. Para operar, toma del SMO, la información de la cáscara de las entidades como así también la constitución de los materiales para obtener los coeficientes de fricción.

Instantánea prescriptiva procedimental

Las instantáneas prescriptivas procedimentales tienen la particularidad que sólo operan sobre las entidades propias del tipo actuador y que además tienen restringidos los movimientos a los que son realizables físicamente por los actuadores. En síntesis son una abstracción para un comando de un actuador de modo que puedan ser integradas con las demás instantáneas presentes en SIET. Las instantáneas prescriptivas procedimentales son generadas por las fuentes coadyuvantes del SA.

Fuente coadyuvante

Una fuente coadyuvante[17] -ver Fuente coadyuvante pág. 55- es un proceso del SA que materializa el movimiento de una entidad generado por una fuente propósito. Cuando el SA funciona varias fuentes coadyuvantes estarán operando en paralelo. Para llevar a cabo su trabajo una fuente coadyuvante recibe las instantáneas prescriptivas declarativas de una entidad en particular generadas por una única fuente propósito. Además de las instantáneas también recibe el requerimiento de la fuente propósito asociado a la entidad objetivo -ver también Fuente coadyuvante pág. 55-. Es función del SIET cada vez que la fuente propósito actualiza el requerimiento notificar del hecho al SA. Con estos datos la fuente coadyuvante selecciona actuadores, determina las configuraciones de hardware para los mismos y genera instantáneas prescriptivas procedimentales.

Luego, la instantáneas procedimentales son pasadas al SIET para su integración el cual responderá al SA si las mismas fueron integradas o no. En el primer caso, el controlador de ejecución -en breve definido- recibe la notificación. En cambio, en el segundo caso la notificación es recibida por la fuente coadyuvante para que regenere las instantáneas procedimentales en función del motivo de la no integración; por ejemplo, si se le informa que una fuente con más importancia tomó el control de los actuadores entonces podría usar otros. En cambio, si no haya otra posibilidad informa al SIET un problema de incapacidad de generar las instantáneas procedimentales.

La fuente coadyuvante generará tantas instantáneas procedimentales como sean necesarias para alcanzar lo declarado en la instantánea declarativa; aunque eso implique adentrarse bastante en el futuro. Eso es así porque es función de la fuente propósito controlar cómo manejar el futuro. Así, puesto que considera toda la acción para llegar a los declarado, es que divide la tolerancia temporal de la instantánea declarativa entre todas las instantáneas procedimentales que genere. Además, la importancia e identidad de las fuentes propósito son incluidas en las instantáneas procedimentales que genere para que el SIET pueda tomar las decisiones durante la integración.

Comando

La definición de comando aquí presentada es naíf debido a las limitaciones del alcance que se estableció para este trabajo. Por eso, un comando es la instrucción a enviar al sistema de gestión de hardware (SGH) que consta de una entidad actuador, una configuración de hardware y en este caso sólo un parámetro que es la velocidad de la rueda. Las entidades actuador son simplemente dos, la rueda derecha y la izquierda.

[17] *Derecho*. Persona que interviene en un proceso sosteniendo la pretensión de una de las partes.

Con la configuración de hardware se selecciona los distintos perfiles de aceleración que configuran el control PID de los actuadores físicos.

Índice de ejecución

La función del índice de ejecución es medir nivel de inanición[18] de las fuentes propósito. Las fuentes expresan sus necesidades temporales por la tolerancia temporal que asignan a sus instantáneas prescriptivas declarativas. Este índice en una instantánea prescriptiva declarativa mide relativamente el instante dentro de la tolerancia temporal de la misma en el cual la instantánea fue ejecutada.
Por ejemplo si la tolerancia temporal para una instantánea prescriptiva declarativa es [400, 800] y la instantánea es transformada en comportamiento en el instante 700, el valor del índice será -0.5 que se obtiene de la siguiente expresión:

$$\frac{(\text{límite\_superior\_tolerancia} + \text{límite\_inferior\_tolerancia}) / 2 - \text{instante}}{\text{límite\_superior\_tolerancia} - (\text{límite\_superior\_tolerancia} + \text{límite\_inferior\_tolerancia}) / 2}$$

Como puede verse de la expresión para todas las instantáneas ejecutadas dentro del período de la tolerancia su índice de ejecución estará entre –1 y +1 mientras que aquellas que se venzan tendrán uno menor a -1 y se asume que no se realizan ejecuciones antes de tiempo por lo tanto no habrá con mayores a 1.
El índice de ejecución para un recorrido es el promedio del índice de ejecución de las últimas n instantáneas prescriptivas declarativas. El valor de n es un parámetro del sistema robótico definido con anterioridad. El índice es computado cuando el controlador de la ejecución -en breve definido- de la fuente propósito recibe la confirmación de la ejecución de los comandos de las instantáneas procedimentales que materializaron la instantánea declarativa.
Recapitulando el índice de ejecución de un recorrido es un valor relativo con centro en el 0. Los valores positivos cercanos a 1 informan de recorridos consistentemente satisfechos y a la inversa los negativos cercanos a -1 de recorridos permanentemente al límite de su período de tolerancia y por último aquellos con IM menores a -1 indican recorridos que están en inanición.

Controlador de ejecución

La función del controlador de ejecución es construir los comandos para el SGH a partir de la notificación enviada por el SIET de que las instantáneas procedimentales fueron integradas, enviar los comandos al SGH y esperar por su conclusión para computar el índice de ejecución. También se encarga de controlar la dependencia entre instantáneas procedimentales. Tal como se describió en la sección del SIET, durante el proceso de integración puede observarse que dos o más fuentes coadyuvantes necesiten un mismo movimiento sobre la misma entidad actuador. En ese caso en vez de declararse un conflicto y que la fuente con menos importancia pierda la capacidad de acción se hace que las instantáneas de la fuente de menos importancia se lleven a cabo con los comandos que surjan de las instantáneas de la fuente de mayor importancia. Es por eso que el controlador de ejecución en el momento que debiera generar los comandos para las instantáneas dependientes no los generará y, cuando regresen del SGH las respuestas de los comandos de las instantáneas independientes, las vinculará tanto a las instantáneas independientes que les corresponde como así también a las instantáneas dependientes de las primeras. De ese modo, ambas fuentes podrán computar adecuadamente el índice de ejecución.

[18] Por inanición se refiere al hecho que una instantánea sea constantemente relegada para la ejecución hasta que finalmente se venza con lo cual la fuente no progresa nunca por no tener suficiente importancia.

El orden de ejecución de los comandos queda determinado por la restricción de ejecución que finalmente le quedó a cada una de las instantáneas procedimentales cuando fueron integradas.
En en SA habrá tantos procesos para el controlador de ejecución como fuente propósitos estén involucradas en ese momento en el SA. Esto es, en vez de como las fuentes coadyuvantes que existirá un proceso por cada par fuente propósito – entidad objetivo los procesos de los controladores de ejecución se corresponden sólo con las fuentes propósito.

Funcionamiento general

El funcionamiento general del sistema de actuación se basa en un proceso distribuidor que distribuye el trabajo y de varios procesos para las fuentes coadyuvantes y los controladores de ejecución. Otras unidades de procesamiento son los motores de dinámica y de detección de contacto. A continuación se describe el distribuidor y el funcionamiento de una fuente coadyuvante.

Distribuidor

La función del proceso distribuidor del SA es:

- Mantener el contexto de acción actualizado en función de los mensajes recibidos del SIET.
- Mantener la información mecánica de las entidades actualizadas para los motores de dinámica y de detección de contacto.
- Recibir desde el SIET las instantáneas prescriptivas declarativas -tanto de entidades no propias como propias-.
- Crear una fuente coadyuvante cuando se reciba la primera instantánea prescriptiva declarativa de un par fuente propósito - entidad objetivo.
- Crear un controlador de ejecución la primera vez que se genera una instantánea procedimental para una fuente propósito.
- Terminar con las fuentes coadyuvantes después de un período prudencial sin recibir una instantánea declarativa para la entidad que gestiona.
- Terminar con los controladores de ejecución que no tienen pendientes notificaciones de prescriptivas procedimentales y que no tenga fuentes coadyuvantes activas.
- Mantener al corriente al SIET sobre cuales son las fuentes coadyuvantes activas.
- Enviar las prescriptivas declarativas a la fuente coadyuvante que le corresponda.
- Enviar al controlador de ejecución que corresponda la notificación de integración recibido desde el SIET.

Comportamiento de fuente coadyuvante

Aquí se presenta una versión muy simplificada del comportamiento de una fuente coadyuvante del SA que se basa en el hecho que operará con un robot diferencial de dos ruedas. Por lo tanto, sólo tendrá dos únicos actuadores y muchas decisiones complejas que debería llevar a cabo una fuente coadyuvante general aquí están ausentes.

1. Solicitar al SIET información descriptiva sobre la entidad objetivo.
2. Obtener las masa de la entidad objetivo.
3. Recibir un flujo de instantáneas declarativas de la entidad objetivo provenientes de una misma fuente propósito.
4. Para cada instantánea prescriptiva declarativa...

(a) Obtener la información de instantánea descriptiva de la entidad objetivo.
(b) Calcular el vector diferencia entre la posición descriptiva y prescriptiva de la entidad objetivo.
(c) En función del vector, la tolerancia temporal, las requerimientos de tolerancia las dimensiones espaciales y las masas involucradas llevar a cabo los siguientes procesos:
    i. Seleccionar los actuadores a utilizar. En esta versión naíf: siempre son las ruedas.
    ii. Seleccionar el mecanismo de sensado sobre los actuadores. Aquí trivialmente se establece el sensado de propiocepción con el SGH.
    iii. Seleccionar las configuraciones de hardware...
        A. En función de la distancia entre Self y la entidad objetivo se establece si el robot previamente se tiene que acercar a la misma. En función de la tolerancia temporal y la distancia a recorrer se elige el perfil de aceleración -configuración de hardware-
        B. Si no puede elegir una configuración de hardware envía el mensaje al SIET que no puede materializar.
        C. Generar una secuencia de instantáneas procedimentales para que el robot se acerque a la entidad objetivo. Las envía al SIET.
        D. Cuando el robot esté cerca re analiza la selección de la configuración de hardware.
        E. Si no puede seleccionar una, envía el mensaje de que no pudo materializar.
        F. Generar instantáneas prescriptivas procedimentales en base a un comportamiento reactivo para empujar la entidad objetivo hasta la posición solicitada en la prescriptiva declarativa. Las envía al SIET.
        G. Si no puede cumplir con los requerimientos de la instantánea declarativa el comportamiento reactivo se detiene y envía el SIET un error de que no puede materializar a la instantánea declarativa.

CAPÍTULO 4 - CASO DE ESTUDIO

Objetivo

En una cancha con paredes y dos arcos hacer que el robot empuje a la pelota dentro el algún arco. La pelota y el robot pueden estar en cualquier sitio de la cancha fuera de los arcos. El robot tendrá 45 segundos para hacer el gol y luego ir a detenerse en el centro de la cancha. Si no lo logra girará tres veces sobre sí mismo y se detendrá.

Ejemplo de código[19]

```
caracteristic color physical
    enum ad_hoc [amarillo, verde, otro];
end

caracteristic luminosidad physical
    enum ad_hoc [blanco, gris, negro];
end

caracteristic IR physical
    enum ad_hoc [brillante, destacado, claro, tenue, oscuro];
end

caracteristic masa physical
    integer grams [0, 10000];
end

caracteristic estadoJuego abstract
    enum ad_hoc [triunfo, derrota, jugando];
end

caracteristic gol abstract
    boolean ad_hoc;
end

caracteristic angulo abstract
    float grade;
end

prototype pelota concrete
    element pelota concrete
        importNURBS('dataNurbsPelota');
        luminosidad fix blanco;
        color fix amarillo;
        masa fix 54 grams;
    end
    construction
        instance for element pelota
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
```

[19] Por convención tácita los prototipos y evoluciones se escriben con la primera letra en minúscula y las entidades y recorridos con la primera letra en mayúscula.

```
            roll 0 grade;
            yaw 0 grade;
        end
    end
  end
end

prototype cancha concrete
  element pared concrete
    importNURBS('dataNurbsPared');
    luminosidad fix gris;
  end
  element arco concrete
    importNURBS('dataNurbsArco');
    luminosidad fix negro;
    masa fixed 10000 grams;
  end
  element centro concrete
    importNURBS('dataNurbsCentro');
    luminosidad fix blanco;
  end
  element cesped concrete
    importNURBS('dataNurbsCesped');
    luminosidad fix gris;
    color fix verde;
  end

  construction
    instance Pared for element pared
      spatial restriction
        course 0 grade;
        distance 0 mm;
        height 0 grade;
        pitch 0 grade;
        roll 0 grade;
        yaw 0 grade;
      end
      masa fixed 10000 grams;
    end
    instance Propio for element arco
      spatial restriction
        course 270 grade;
        distance 52.5 mm;
        height 0 grade;
        pitch 0 grade;
        roll 0 grade;
        yaw 0 grade;
      end
    end
    instance Adversario for element arco
      spatial restriction
        course 90 grade;
        distance 52.5 mm;
        height 0 grade;
        pitch 0 grade;
        roll 0 grade;
        yaw 0 grade;
```

```
            end
        end
        instance Centro for element centro
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
        instance Cesped for element cesped
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
    end
end

prototype self concrete
    element carcasa concrete
        importNURBS('dataNurbsCarcasa');
    end
    element rueda concrete
        importNURBS('dataNurbsRueda');
    end
    element sensorIR concrete
        importNURBS('dataNurbsSensorIR');
    end

    prototype camara concrete
        element camara concrete
            importNURBS('dataNurbsCamara');
        end
        element luminosidad virtual
            importNURBS('dataNurbsLuminosidad');
        end
        element color virtual
            importNURBS('dataNurbsColor');
        end
        construction
            instance detector for element camara is sensor
                spatial restriction
                    course 0 grade;
                    distance 0 mm;
                    height 0 grade;
                    pitch 0 grade;
                    roll 0 grade;
                    yaw 0 grade;
                end
            end
```

```
        instance luminosidad for element luminosidad is range
            spatial restriction
                course 0 grade;
                distance 25 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
        instance color for element color is range
            spatial restriction
                course 0 grade;
                distance 25 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
    end
end

prototype sensorIRLI concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIRLI');
    end
    construction
        instance detector for element sensorIR is sensor
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
        instance alcance_1 for element alcance_1 is range
            spatial restriction
                course 0 grade;
                distance 55 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
    end
end

prototype sensorIROI concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIROI');
    end
    construction
        instance detector for element sensorIR is sensor
```

```
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
        instance alcance_1 for element alcance_1 is range
            spatial restriction
                course 0 grade;
                distance 55 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
    end
end

prototype sensorIRFI concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIRFI');
    end
    construction
        instance detector for element sensorIR is sensor
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
        instance alcance_1 for element alcance_1 is range
            spatial restriction
                course 0 grade;
                distance 55 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
    end
end

prototype sensorIRFD concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIRFD');
    end
    construction
        instance detector for element sensorIR is sensor
            spatial restriction
```

```
            course 0 grade;
            distance 0 mm;
            height 0 grade;
            pitch 0 grade;
            roll 0 grade;
            yaw 0 grade;
        end
    end
    instance alcance_1 for element alcance_1 is range
        spatial restriction
            course 0 grade;
            distance 55 mm;
            height 0 grade;
            pitch 0 grade;
            roll 0 grade;
            yaw 0 grade;
        end
    end
end

prototype sensorIROD concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIROD');
    end
    construction
        instance detector for element sensorIR is sensor
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
        instance alcance_1 for element alcance_1 is range
            spatial restriction
                course 0 grade;
                distance 55 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
    end
end

prototype sensorIRLD concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIRLD');
    end
    construction
        instance detector for element sensorIR is sensor
            spatial restriction
                course 0 grade;
```

```
            distance 0 mm;
            height 0 grade;
            pitch 0 grade;
            roll 0 grade;
            yaw 0 grade;
        end
    end
    instance alcance_1 for element alcance_1 is range
        spatial restriction
            course 0 grade;
            distance 55 mm;
            height 0 grade;
            pitch 0 grade;
            roll 0 grade;
            yaw 0 grade;
        end
    end
end

prototype sensorIRTD concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIRTD');
    end
    construction
        instance detector for element sensorIR is sensor
            spatial restriction
                course 0 grade;
                distance 0 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
        instance alcance_1 for element alcance_1 is range
            spatial restriction
                course 0 grade;
                distance 55 mm;
                height 0 grade;
                pitch 0 grade;
                roll 0 grade;
                yaw 0 grade;
            end
        end
    end
end

prototype sensorIRTI concrete
    element alcance_1 virtual
        importNURBS('dataNurbsIRTI');
    end
    construction
        instance detector for element sensorIR is sensor
            spatial restriction
                course 0 grade;
                distance 0 mm;
```

```
            height 0 grade;
            pitch 0 grade;
            roll 0 grade;
            yaw 0 grade;
        end
    end
    instance alcance_1 for element alcance_1 is range
        spatial restriction
            course 0 grade;
            distance 55 mm;
            height 0 grade;
            pitch 0 grade;
            roll 0 grade;
            yaw 0 grade;
        end
    end
end

construction
    instance Carcasa for element carcasa is pasive
        spatial restriction
            course 0 grade;
            distance 0 mm;
            height 0 grade;
            pitch 0 grade;
            roll 0 grade;
            yaw 0 grade;
        end
    end
    instance RuedaDerecha for element rueda is actuator
        spatial restriction
            course 90 grade;
            distance 23 mm;
            height -28 grade;
            pitch 0 grade;
            roll 90 grade;
            yaw unlimint ;
        end
    end

    instance RuedaIzquierda for element rueda is actuator
        spatial restriction
            course -90 grade;
            distance 23 mm;
            height -28 grade;
            pitch 0 grade;
            roll -90 grade;
            yaw unlimint;
        end
    end
    instance Camara for prototype camara is sensor
        spatial restriction
            course 0 grade;
            distance 15 mm;
            height 90 grade;
            pitch 0 grade;
```

```
        roll 0 grade;
        yaw 0 grade;
    end
end
instance IRLateralIzquierdo for prototype sensorIRLI is sensor
    spatial restriction
        course -90 grade;
        distance 25 mm;
        height 0 grade;
        pitch 0 grade;
        roll -90 grade;
        yaw 0 grade;
    end
end
instance IROblicuoIzquierdo for prototype sensorIROI is sensor
    spatial restriction
        course -45 grade;
        distance 25 mm;
        height 0 grade;
        pitch 0 grade;
        roll -45 grade;
        yaw 0 grade;
    end
end
instance IRFrontalIzquierdo for prototype sensorIRFI is sensor
    spatial restriction
        course -10 grade;
        distance 25 mm;
        height 0 grade;
        pitch 0 grade;
        roll 0 grade;
        yaw 0 grade;
    end
end
instance IRFrontalDerecho for prototype sensorIRFD is sensor
    spatial restriction
        course 10 grade;
        distance 25 mm;
        height 0 grade;
        pitch 0 grade;
        roll 0 grade;
        yaw 0 grade;
    end
end
instance IROblicuoDerecho for prototype sensorIROD is sensor
    spatial restriction
        course 45 grade;
        distance 25 mm;
        height 0 grade;
        pitch 0 grade;
        roll 45 grade;
        yaw 0 grade;
    end
end
instance IRLateralDerecho for prototype sensorIRLD is sensor
    spatial restriction
        course 90 grade;
```

```
            distance 25 mm;
            height 0 grade;
            pitch 0 grade;
            roll 90 grade;
            yaw 0 grade;
        end
    end
    instance IRTraseroDerecho for prototype sensorIRTD is sensor
        spatial restriction
            course 170 grade;
            distance 25 mm;
            height 0 grade;
            pitch 0 grade;
            roll 180 grade;
            yaw 0 grade;
        end
    end
    instance IRTraseroIzquierdo for prototype sensorIRTI is sensor
        spatial restriction
            course -170 grade;
            distance 25 mm;
            height 0 grade;
            pitch 0 grade;
            roll x grade;
            yaw -180 grade;
        end
    end
end
binding
    sensingzone
        sensor this.Camara.Detector;
        hardwareconfiguration 100;
        caracteristic Luminosidad;
        range this.Camara.Luminosidad;
        resolution  (course, {(90%,{blanco, gris, negro})})
                    (height, {(90%,{blanco, gris, negro})});
    end
    sensingzone
        sensor this.Camara.Detector;
        hardwareconfiguration 110;
        caracteristic Color;
        range this.Camara.Color;
        resolution  (course, {(90%,{amarillo, verde, otro})})
                    (height, {(90%,{amarillo, verde, otro})});
    end
    sensingzone
        sensor this.IRLateralIzquierdo.Detector;
        hardwareconfiguration 1;
        caracteristic IR;
        range this.IRLateralIzquierdo.alcance_1;
        resolution  (distance,  {(80%,{brillante})
                                 (70%,{destacado, claro})
                                 (60%,{tenue})
                                 (50%,{oscuro})
                                }
                    );
    end
```

```
sensingzone
    sensor this.IROblicuoIzquierdo.Detector;
    hardwareconfiguration 2;
    caracteristic IR;
     range this.IROblicuoIzquierdo.alcance_1;
    resolution  (distance,  {(80%,{brillante})
                             (70%,{destacado, claro})
                             (60%,{tenue})
                             (50%,{oscuro})
                            }
                );
end
sensingzone
    sensor this.IRFrontalIzquierdo.Detector;
    hardwareconfiguration 3;
    caracteristic IR;
     range this.IRFrontalIzquierdo.alcance_1;
    resolution  (distance,  {(80%,{brillante})
                             (70%,{destacado, claro})
                             (60%,{tenue})
                             (50%,{oscuro})
                            }
                );
end
sensingzone
    sensor this.IRFrontalDerecho.Detector;
    hardwareconfiguration 4;
    caracteristic IR;
     range this.IRFrontalDerecho.alcance_1;
    resolution  (distance,  {(80%,{brillante})
                             (70%,{destacado, claro})
                             (60%,{tenue})
                             (50%,{oscuro})
                            }
                );
end
sensingzone
    sensor this.IROblicuoDerecho.Detector;
    hardwareconfiguration 5;
    caracteristic IR;
     range this.IROblicuoDerecho.alcance_1;
    resolution  (distance,  {(80%,{brillante})
                             (70%,{destacado, claro})
                             (60%,{tenue})
                             (50%,{oscuro})
                            }
                );
end
sensingzone
    sensor this.IRLateralDerecho.Detector;
    hardwareconfiguration 6;
    caracteristic IR;
     range this.IRLateralDerecho.alcance_1;
    resolution  (distance,  {(80%,{brillante})
                             (70%,{destacado, claro})
                             (60%,{tenue})
                             (50%,{oscuro})
```

```
                                    }
                        );
        end
        sensingzone
            sensor this.IRTraseroDerecho.Detector;
            hardwareconfiguration 7;
            caracteristic IR;
             range this.IRTraseroDerecho.alcance_1;
            resolution  (distance,  {(80%,{brillante})
                                     (70%,{destacado, claro})
                                     (60%,{tenue})
                                     (50%,{oscuro})
                                    }
                        );
        end
        sensingzone
            sensor this.IRTraseroIzquierdo.Detector;
            hardwareconfiguration 8;
            caracteristic IR;
             range this.IRTraseroIzquierdo.alcance_1;
            resolution  (distance,  {(80%,{brillante})
                                     (70%,{destacado, claro})
                                     (60%,{tenue})
                                     (50%,{oscuro})
                                    }
                        );
        end
    end
end

prototype metegolSolo abstract
    gol default False;
    estadoJuego default jugando;
end

prototype anguloRotacion abstract
    angulo default 0.0 grade;
end

evolution prescriptive main
    context around Cancha
        cast cancha.(new):Cancha;
        cast pelota.(new):Pelota;
        cast metegolSolo.(new):MetegolSolo;
        cast anguloRotacion.(new):AnguloRotacion;
        travel acercar:AcercarseA;
        travel empujarHacia:EmpujarHacia;
        travel llevarJuego:LlevarJuego;
        travel rotar:DanzaLuto;
    end context

    requirement
        feature descriptive Pelota:pelota_burda is
            entity.confiability: 75 confidence;
            distance.confiability: 75 confidence;
            distance.tolerance: 30 mm;
            course.confiability: 75 confidence;
```

```
        course.tolerance: 30 grade;
    end feature
    feature prescriptive Pelota:pelota_detalle is
        entity.confiability: 90 confidence;
        distance.confiability: 90 confidence;
        distance.tolerance: 7 mm;
        course.confiability: 90 confidence;
        course.tolerance: 6 grade;
    end feature
    feature descriptive Arcoadversario:arco_gambeta is
        entity.confiability: 90 confidence;
        distance.confiability:75 confidence;
        distance.tolerance: 20 mm;
        course.confiability: 75 confidence;
        course.tolerance: 6 grade;
    end feature

    style busqueda primer
        pelota_burda;
    end style

    style gambeta
        pelota_detale;
        arco_gambeta;
    end style
end requirement

produce on
    reaction velocity 0.5 hz
    planification velocity 1
    horizon 1 seconds
    delta time 25 %
    delta processtime 25 %
    for
    start
        LlevarJuego(command:start);
        EmpujarHacia(command:start,
                        Objeto:Pelota,
                        Destino:Cancha.Arcodversario);
        AnguloRotacion.angulo = 1080.0 grade;
    end start
    while
        switch milestone
            case [distance(Self, Pelota) >= 45 mm]
                changestyle busqueda;
            end case
            case [distance(Self, Pelota) < 45 mm]
                changestyle gambeta;
            end case
        end switch
    end while
    resource
      // Se llegó a Main sin haber resuelto el problema. Se detiene el
      robot.
        Main(command: stop );
    end resource
end produce
```

```
syncronization
    about EmpujarHacia
        trigger gol
            situation around Cancha [MetegolSolo.gol]
            when first_time
            fire carreraFinal
        end trigger
        trigger horaFinal
            situation around Cancha
                [MetegolSolo.estado = derrota]
            when first_time
            fire finalVencido1
        end trigger
    end about

    about  AcercarseA
        trigger gano
            situation around Cancha
                [MetegolSolo.estado = triunfo]
            when first_time
            fire finalTriunfal
        end trigger
        trigger horaFinal
            situation around Cancha
                [MetegolSolo.estado = derrota]
            when first_time
            fire finalVencido2
        end trigger
    end about

    transition finalTriunfal
        over (AcercarseA.gano)
        do
            arrange(    conclude(AcercarseA.gano),
                        Main(command: stop )
                    );
    end transition

    transition carreraFinal
        over (EmpujarHacia.gol)
        do
            arrange(    conclude(EmpujarHacia.gol),
                        lever(
                            AcercarseA(command:start ,
                            Objetivo:CentroCancha))
                    );
    end transition

    transition finalVencido1
        over (EmpujarHacia.horaFinal)
        do
            arrange(conclude(EmpujarHacia.horaFinal),
                    lever(DanzaLuto(comand:start, Marco:Cancha,
                                Control:AnguloRotacion))
                    );
    end transition
```

```
        transition finalVencido2
            over (AcercarseA.horaFinal)
            do
                arrange(conclude(AcercarseA.horaFinal),
                        lever(DanzaLuto(comand:start, Marco:Cancha,
                                Control:AnguloRotacion))
                        );
        end transition
    end syncronization
end evolution

evolution descriptive pelota
    context around Cesped
        starring pelota.(choose):Pelota;
        guest .(where type = concrete):Cesped;
        guest .(where type = concrete
                and distance(this, Pelota) < 100 mm):Obstaculo[];
    end context

    produce on
        reaction velocity 10 hz
        planification velocity 2
        horizon 2 seconds
        delta time 25 %
        delta processtime 25 %
        for
        while
            perform on  Pelota
            do
                // Utiliza el motor de dinámica para calcular la posición de la
                pelota.
                position = dynamic(Pelota, Obstaculo[]);
            end perform
        end while
    end produce
end evolution

evolution prescriptive llevarJuego
    context around Estadio
        starring metegolSolo.(choose):MetegolSolo;
        cast cancha.(choose):Estadio;
        cast pelota.(choose):Pelota;
    end context

    produce on
        reaction velocity 2 hz
        planification velocity 1
        horizon 2 seconds
        delta time 25 %
        delta processtime 25 %
        for
        start
            watch tiempolimite.start(45 seconds);
        end start
        while
            perform on
```

```
            do
                if in(Pelota, Estadio.RedArco) and not tiempolimite.over
                then
                    gol = true;
                end if;
                if in(Self, Estadio.CentroCancha) and gol
                    and not tiempolimite.over
                then
                    estado = triunfo;
                end if;
                if tiempolimite.over and not in(Self, Estadio.CentroCancha)
                then
                    estado = derrota;
                end if;
            end perform
        end while
    end produce
end evolution

evolution prescriptive acercar
    context around Objetivo
        starring self.(Self):Self;
        guest .(where type in(concrete, virtual)): Objetivo;
        guest .(where type = concrete
                    and distance(this, Pelota) < 100 mm
                    and trayectory(this, Pelota):Obstaculo;
        travel seguir_contorno:RodearObstaculo;
    end context

    produce on
        reaction velocity 4 hz
        planification velocity 3
        horizon 3 seconds
        delta time 25 %
        delta processtime 25 %
        for
        while
            perform on Self
            do
                if distance < 3 mm then
                    distance = 0 mm;
                elseif distance >= 3 mm then
                    distance = distance – 3 mm;
                end
                if abs(course) < 6 grade then
                    course = 0 grade;
                elseif abs(course) >= 6 grade then
                    course = course – sgn(course) * 6 grade;
                end
            end perform
        end while
    end produce

    syncronization
    about this
            trigger deteccion_obstaculo_proximo
                situation around Objetivo
```

```
                    [surfacedistance(Self, Obstaculo) < 20 mm and
                     decresing(surfacedistance(Self, Obstaculo))]
                  // está muy próximo al obstáculo y acercándose
            when first-time
            fire esquivar_obstaculo
         end trigger
         trigger mantener_rodeo_obstaculo
            situation around Objetivo
                    [surfacedistance(Self, Obstaculo) < 20 mm and
                    NoDecresing(surfaceDistance(Self, Objetivo))]
                  // está próximo al obstáculo pero no se acerca al
                  objetivo
            when each-time
            fire rodeando
         end trigger
         trigger finalizar_rodeo_obstaculo
            situation around Objetivo
                      [surfacedistance(Self, Obstaculo) < 20 mm and
                      Decresing(surfaceDistance(Self, Objetivo))]
                // está próximo al obstáculo y se acercó al objetivo
            when each-time
            fire final_rodeo
         end trigger
      end about
      about RodearObstaculo
         trigger objetivo_mas_cerca
            situation around Objetivo
                  [Decresing(surfaceDistance(Self, Objetivo))]
               // Se está acercando al objetivo. Quizás es el momento de
               abandonar el seguir la pared.
            when first-time
            fire final_rodeo
         end trigger
      end about

      transition esquivar_obstaculo
         over ( this.deteccion_obstaculo_proximo )
            arrange(    inhibit(this.deteccion_obstaculo_proximo),
                        lever(RodearObstaculo(command:start,
                                    Objetivo:Obstaculo)
                              )
                    );
      end transition
      transition rodeando
         over ( this.mantener_rodeo_obstaculo)
            inhibit(this.mantener_rodeo_obstaculo);
      end transition
      transition final_rodeo
         over (    RodearObstaculo.objetivo_mas_cerca,
                   this.finalizar_rodeo_obstaculo )
            arrange(    conclude(RodearObstaculo.objetivo_mas_cerca),
                        this.finalizar_rodeo_obstaculo);
      end transition
   end syncronization
end evolution

evolution prescriptive empujarHacia
```

```
context around Destino
    starring self.(Self):Self;
    guest pelota.(where diametre between Self.diametre * 0.35
                                and Self.diametre * 1.75):Objeto;
                        // No intenta empujar otras cosas que no sean pelotas;
    guest .(where type in(concrete, virtual)): Destino;
    cast punto.( where
              course = Objeto.course and
              distance = Objeto.distance
                              + Objeto.diametre / 2):PuntoDeAplicacion;
    travel acercar:IrA;
end context
produce on
    reaction velocity 8 hz
    planification velocity 1
    horizon 1 seconds
    delta time 25 %
    delta processtime 25 %
    for
    while
        perform on Self
            do
            if distance(Self, PuntoDeAplicacion) <= 30 mm then
                if  abs(Self.course –
                          PuntoDeAplicacion.course) <= 28 grade then
                        roll = 0;
                        course = PuntoDeAplicacion.course;
                        distance = PuntoDeAplicacion.distance – 7 mm;
                else
                        roll = 0;
                        course = PuntoDeAplicacion.course;
                        distance = PuntoDeAplicacion.distance;
                end
            end
        end perform
    end while
end produce
syncronization
    about this
        trigger lejos_de_objeto
            situation around Destino
                [distance(Self, PuntoDeAplicacion) > 30 mm]
            when first-time
            fire comezar_aproximacion
        end trigger
    end about
    about IrA
        trigger llego_a_destino
            situation around Destino
                [distance(Self, PuntoDeAplicacion) < 3 mm ]
            when first-time
            fire detener_aproximacion
        end trigger
    end about

    transition comenzar_aproximacion
        over (this.lejos_de_objeto)
```

```
            arrange(inhibit(this.lejos_de_objeto),
                lever(IrA(commad:start, Objetivo:PuntoDeAplicacion));
        end transition

        transition detener_aproximacion
            over (IrA.llego_a_destino)
                conclude(IrA.llego_a_destino);
        end transition
    end syncronization
end evolution

evolution prescriptive seguirContorno
    context around Objetivo
        starring self.(Self):Self;
        guest .(where type = concrete): Objetivo;
    end context
    produce on
        reaction velocity 4 hz
        planification velocity 3
        horizon 3 seconds
        delta time 25 %
        delta processtime 25 %
        for
        while
            perform on Self
            do
                if surfacedistance(Self) < 13 mm then
                    distance = Self.distance + 3 mm;
                else surfacedistance(Self) > 18 mm then
                    distance = Self.distance – 3 mm;
                end
                course = Self.course + 6 grade;
            end perform
        end while
    end produce
end evolution

evolution prescriptive rotar
    state
        float diferenciaAngulo grade;
    end state
    context around Marco
        starring self.(Self):Self;
        guest .(where type = concrete): Marco;
        guest .(where prototype = anguloRotacion): Control;
    end context
    produce on
        reaction velocity 4 hz
        planification velocity 3
        horizon 3 seconds
        delta time 25 %
        delta processtime 25 %
        for
        start
             diferenciaAngulo = Self.course +  Control.angulo;
        end start
        while
```

```
            perform on Self
            do
                if abs(Self.course -  diferenciaAngulo) > 18 grade then
                    course = Self.course + 6 grade;
                end if
            end perform
        end while
    end produce
end evolution
```

CAPÍTULO 5 - CONCLUSIÓN

Luego de haber podido diseñar un framework para el software de un robot autónomo en base a las guías planteadas por la arquitectura propuesta se concluye que es factible proseguir con la codificación de un prototipo funcional del antedicho software. Además del prototipo se propone el siguiente camino para la conformación de un robot autónomo avanzado. Así, los próximos objetivos derivados de este trabajo son:

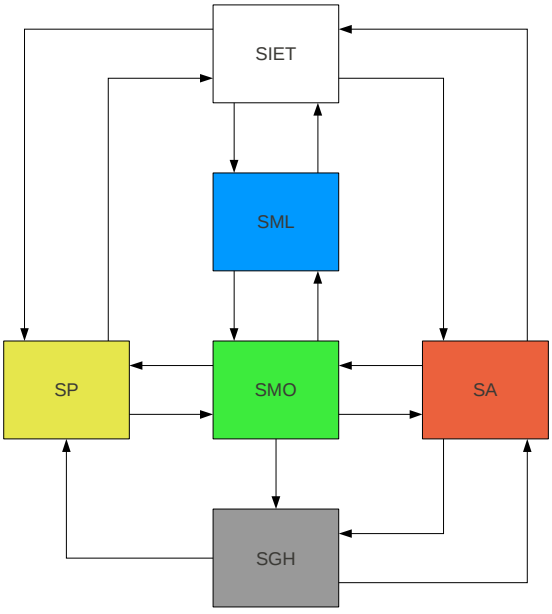
1. Implementar un prototipo funcional del framework para un robot de un ambiente simulado para validar su funcionamiento.
2. Desarrollar el lenguaje reflexivo con la intención de que el robot pueda auto programarse bajo ciertas circunstancias.
3. Agregar un sistema para creación automática de prototipos de entidades.
4. Generalizar el diseño para incluir robots más allá del tipo de dirección diferencial.
5. Agregar un sistema de interacción social primitivo -dejar de ser sólo *Self*-.
6. Agregar un sistema de imitación de comportamiento y uso de herramientas.

APÉNDICES

Interfaces entre sistemas

Los distintos sistemas de la arquitectura se comunican entre sí a través de mensajes asincrónicos. Se presenta para cada sistema y en orden alfabético los mensajes que envía y que recibe incluyéndose una breve explicación, la signatura del mensaje y las referencias cruzadas de que sistemas lo utilizan.
Los mensajes se clasifican en tres tipos:

- solicitudes: son mensajes que esperan una respuesta.
- respuestas: son mensajes que responden a una solicitud.
- notificaciones: son mensajes que informan algo sin haber sido solicitado ni esperan una respuesta.



Descripción de la interfaz del SIET

Mensajes recibidos por el SIET

Campo -solicitud-

(id fuente, nro. de mensaje, id entidad virtual)
Enviado por: SP
Propósito: que el SP pueda verificar dentro de una región espacial específica denotada por la entidad virtual cuáles entidades se hallan y en qué posiciones.

Crear entidad -solicitud-

(id fuente, nro de mensaje, id prototipo)
Este mensaje es enviado por los sistemas que necesitan disponer de una entidad que no existe previamente en el modelo del SIET. Una vez que la entidad existe se pueden comenzar a realizar pedidos por

las instantáneas de las mismas o enviar instantáneas prescriptivas para establecer su posición.

Enviado por: SML

Propósito: que los recorridos puedan crear las entidades que necesitan en el modelo a partir de los prototipos.

Enviado por: SP

Propósito: que el SP pueda crear entidades virtuales en el modelo para usar el SIET como una memoria espacio temporal para llevar a cabo el sensado.

Desarrollo de aplicación fallido -notificación-

(id instantánea, error)

Enviado por: SA

Propósito: notificar al SIET cuando el SA no se puede generar un desarrollo de instantáneas prescriptivas procedimentales que satisfaga lo necesario de una instantánea prescriptiva seleccionada por el SIET.

Entidades en modelo -solicitud-

(id fuente, nro de mensaje, expresión del lenguaje de consulta)

Enviado por: SML

Propósito: Que los recorridos del SML puedan en la declaración de entidades verificar la existencia de una entidad en el modelo antes de crear una nueva instancia.

Enviado por: SP

Propósito: Obtener las entidades virtuales de tipo *range* que están activas en cada momento junto con las entidades sensor a la cuales están asociadas.

Establecer característica -notificación-

(id fuente, id entidad, id característica, valor)

Enviado por: SML

Propósito: Permitir que los recorridos establezcan valores en las características variables de las entidades que utilizan.

Enviado por: SP

Propósito: Almacenar las características variables sensadas en las instantáneas de las entidades.

Instantánea descriptiva -respuesta-

(id entidad, instantánea{id instantánea, tipo, etc.})

Enviado por: SP

Propósito: Integrar en el modelo las instantáneas descriptivas presentes de las entidades que esté sensando el SP en respuesta a los requerimientos del SIET.

Instantánea destruida -notificación-

(id instantánea)

Enviado por: SP

Propósito: Eliminar del modelo una instantánea prescriptiva del futuro previamente notificada por el SP ya sea porque aquel futuro ya no es posible o porque ya no es más necesitado.

Enviado por: SML

Propósito: Eliminar del modelo una instantánea previamente notificada por un recorrido ya que no tiene más sentido su existencia para aquel.

Enviado por: SA

Propósito: Eliminar del modelo una instantánea prescriptiva propia previamente notificada por el SA dado que ya no es posible operar sobre la entidad recurso en cuestión o porque se asignó el recurso a otro fin.

Instantánea modificada -notificación-

(conjunto de restricción, instantánea{id instantánea, tipo, etc.}[, id configuración de hardware])

Enviado por: SP

Propósito: Ajustar el plan de movimiento del sensado activo de un modo más eficiente que destruyendo y creando instantáneas muy similares. En este caso es válido el parámetro configuración de hardware.

Enviado por: SML

Propósito: Actualizar el plan del recorrido -en caso que haga uso de esto- de un modo más eficiente que destruyendo y creando instantáneas muy similares.

Enviado por: SA

Propósito: Actualizar el plan de movimiento de entidades propias de tipo actuador un modo más eficiente que destruyendo y creando instantáneas muy similares. En este caso es válido el parámetro configuración de hardware.

Instantánea producida -notificación-

(id fuente, conjunto de restricción, instantánea{id instantánea, id entidad, tipo, etc.} [, id configuración de hardware}])

El atributo “conjunto de restricción” contiene el conjunto de identificadores de instantáneas –posiblemente vacío- que regula la sincronización de la instantánea producida. El atributo “instantánea” es compuesto conteniendo todos los atributos de la instantánea enviada. El id de instantánea es creado por la fuente. Consta del id de sistema fuente, del id de la fuente y un nro de secuencia dentro de la fuente. Así se puede garantizar la unicidad global del identificador.

Enviado por: SP

Propósito: Integrar en el modelo el movimiento de las entidades zona de sensado para el sensado activo y registrar las dimensiones de las entidades virtuales usadas por el SP como memorias espacio temporales. En este caso es válido el parámetro configuración de hardware.

Enviado por: SML

Propósito: Integrar en el SIET las instantáneas producidas por los recorridos.

Enviado por: SA

Propósito: Integrar en el SIET las instantáneas prescriptivas de entidades propias de tipo actuador procedimentales generadas a partir de las instantáneas prescriptivas declarativas. En este caso es válido el parámetro configuración de hardware.

Instantánea -solicitud-

(id fuente, nro de mensaje, id entidad, tipo instantánea)

Este mensaje es enviado por los sistemas que consumen instantáneas del SIET. El "tipo de instantánea" puede ser "descriptiva" o "especulativa".

Enviado por: SML

Propósito: Realizar la solicitud de necesidad de información sobre una entidad utilizada por un recorrido.

Enviado por: SP

Propósito: Realizar las solicitudes de instantáneas especulativas para obtener la información del dominio de aplicación del robot, de la posición de las entidades virtuales del alcance de las zonas de sensado, de las entidades virtuales usadas para registrar las posiciones del espacio ya exploradas y las entidades virtuales del espacio incógnito.

Enviado por: SA

Propósito: Obtener instantáneas descriptivas de las entidades del contexto de acción y de las entidades propias con el fin de recalibrar los sensores de propiocepción.

Fuente activada -notificación-

(id fuente, id entidad de referencia)

Notifica al SIET de una nueva fuente y la entidad de referencia que la misma utiliza.

Enviado por: SP

Propósito: Registrar en el SIET las fuentes coadyuvantes.

Enviado por: SA

Propósito: Registrar en el SIET las fuentes coadyuvantes.

Enviado por: SML

Propósito: Registrar en el SIET los recorridos.

Fuente desactivada -notificación-

(id fuente)

Enviado por: SA

Propósito: Registrar en el SIET las fuentes coadyuvantes que han sido desactivadas.

Enviado por: SP

Propósito: Registrar en el SIET las fuentes coadyuvantes que han sido desactivadas.

Enviado por: SML

Propósito: Registrar en el SIET los recorridos que han sido desactivadas.

Índice de ejecución -notificación-

(id fuente, valor)

Enviado por: SA

Propósito: Notificar al SIET los valores del índice de ejecución para cada fuente.

Liberar entidad -notificación-

(id fuente, id entidad)

Enviado por: SP

Propósito: Liberar el uso de las entidades que no son más necesarias para los procesos del SP.

Enviado por: SA

Propósito: Liberar el uso de las entidades que no son más necesarias para los procesos del SA.

Materialización fallida -notificación-

(id instantánea, error)

Enviado por: SA

Propósito: Informar al SIET los errores producidos por problemas con los recursos; ya sea por falla o por disponibilidad.

Obtener característica -solicitud-

(id fuente, nro de mensaje, id entidad, id característica, id elemento)
Este mensaje se utiliza para solicitar los valores de una característica de una entidad independientemente si la característica está definida como variable o no.
Enviado por: SP
Propósito: Solicitar información de las entidades del modelo del SIET.
Enviado por: SML
Propósito: Solicitar información de las entidades del modelo del SIET.

Operación espacio temporal -solicitud-

(id fuente, nro. de mensaje, operación)
Se envía una operación con parámetros para ser evaluada por el motor de álgebra espacial del SIET. Algunas de las operaciones disponibles son:

- distance (in: id instantánea, in: id instantánea): real
- over (in: id instantánea, in: id instantánea): boolean
- surfacedistance (in: id instantánea, in: id instantánea): real

"operación" es de un tipo de datos especial que lleva en forma eficiente la invocación de la operación para que no tenga que ser parseada en el momento de la resolución. El id sistema emisor es el sistema fuente.
Enviado por: SML
Propósito: Enviar una operación del álgebra espacio temporal al SIET para que la resuelva.
Enviado por: SP
Propósito: Evaluar si la entidades objetivo de los prejuicios están contenidas en las zonas de sensado.

Requerimiento -notificación-

(id fuente, id entidad, requerimiento)
Enviado por: SA
Propósito: Registrar en el SIET los requerimientos de las fuentes coadyuvantes para sus entidades.
Enviado por: SP
Propósito: Registrar en el SIET los requerimientos de las fuentes coadyuvantes para sus entidades.
Enviado por: SML
Propósito: Registrar en el SIET los requerimientos de los recorridos para sus entidades.

Mensajes enviados por el SIET

Arrancar recorrido -notificación-

(nombre evolucion, tipo de evolucion)
Recibido por: SML
Propósito: Arrancar los recorridos especulativos y descriptivos de las entidades.

Campo -respuesta-

(nro. secuencia, conjunto)

Envía un conjunto de tuplas conformadas con los atributos (identificador de entidad, id prototipo, posición con respecto a Self) Las entidades estarán dentro de la entidad virtual recibida en la solicitud -incluyendo la dimensión tiempo-. El conjunto puede ser vacía.

Recibido por: SP

Propósito: Informar lo que para el SIET se encuentra en una región del espacio en ese momento.

Crear entidad -respuesta-

(id fuente, nro de mensaje, id entidad)

Recibido por: SML

Propósito: Crear y devolver el id de la entidad creada con sus confiabilidades de posición nulas y las características variables de la misma indefinidas.

Recibido por: SP

Propósito: Responder con el identificador de la entidad virtual creada.

Detener recorrido -notificación-

(nombre de recorrido)

Recibido por: SML

Propósito: Detener los recorridos cuando no haya más solicitudes de instantáneas a resolver con los recorridos especulativos o descriptivos correspondientes.

Entidad dentro de contexto de acción -notificación-

(id entidad)

Recibido por: SA

Propósito: Informar al motor de dinámica y de detección de colisiones las entidades físicas que debe considerar en sus modelos internos.

Entidad fuera de contexto de acción -notificación-

(id entidad)

Recibido por: SA

Propósito: Informar al motor de dinámica y de detección de colisiones las entidades físicas que debe considerar en sus modelos internos.

Entidades en modelo -respuesta-

(id fuente, nro de mensaje, identificadores de entidades)

Es un mensaje que envía el SIET con la respuesta de que entidades satisfacen el criterio suministrado previamente por el mensaje de solicitud. El atributo “identificadores de entidades” es un conjunto de identificadores de entidad que satisfacen la solicitud previamente recibida. El conjunto puede ser vacío si no existe entidad que satisfaga la solicitud.

Recibido por: SML

Propósito: Obtener si una entidad existe o no.

Recibido por: SP

Propósito: Obtener la capacidad de sensado actual del robot.

Instantánea descriptiva -solicitud-

(id fuente, id entidad, requerimiento, importancia)

Recibido por: SP

Propósito: Iniciar el sensado de la entidad solicitada.

Instantánea fallida -notificación-

(id fuente, id instantánea, error)

Recibido por: SP

Propósito: Informar que una instantánea prescriptiva no pudo ser llevada a cabo.

Recibido por: SML

Propósito: Informar que una instantánea prescriptiva no pudo ser llevada a cabo

Instantánea declarativa -notificación-

(conjunto de restricción, instantánea {id instantánea, tipo, etc.})

El mensaje es enviado por el SIET al SA para informarle la instantánea -sea de una entidad propia o no- para materializar. El atributo "conjunto de restricción" contiene el conjunto de identificadores de instantánea –posiblemente vacío- que regula la sincronización de la instantánea seleccionada y el atributo "instantánea" es compuesto conteniendo todos los atributos de dicha instantánea. Este mensaje puede enviarse más de una vez para una misma instantánea lo cual indica una actualización de los datos asociados a esta.

Recibido por: SA

Propósito: Indicar al SA cuales instantáneas debe materializar.

Instantánea -respuesta-

(id fuente, instantánea{id instantánea, id entidad, tipo, etc.})

Recibido por: SP

Propósito: Recibir la información de las instantáneas especulativas y de las entidades virtuales que representan el alcance de las zonas de sensado activas para ser utilizadas en la confección de la estrategia de sensado del momento.

Recibido por: SML

Propósito: Recibir la información de las instantáneas que proveen información sobre las entidades.

Recibido por: SA

Propósito: Recibir la información de las instantáneas descriptivas que informan la posición de la entidad solicitada.

Obtener característica -respuesta-

(id fuente, nro de mensaje, valor)

Recibido por: SP

Propósito: Obtener el valor de una característica de una entidad.

Recibido por: SML

Propósito: Obtener el valor de una característica de una entidad.

Operación espacio temporal -respuesta-

(id fuente, nro. de mensaje, resolución)

Recibido por: SML

Propósito: Respuesta de la operación del álgebra espacio temporal.

Recibido por: SP

Propósito: Respuesta de la evaluación si las entidades objetivos están o no contenidas en las zonas de sensado.

Tamaño de muestra para índice de ejecución -notificación-

(id fuente, cantidad, período de notificación)

El parámetro *cantidad* informa la cantidad de instantáneas prescriptivas a considerar y el *período de notificación* cada cuanto tiene que interrumpir el SA para informar el resultado.

Recibido por: SA

Propósito: Informa los parámetros utilizados para calcular índices de performance de la actuación.

Umbral de futuro -notificación-

(id fuente, cantidad)

Recibido por: SML

Propósito: Este mensaje informa a los recorridos hasta cuando en el futuro es conveniente planificar.

Descripción de la interfaz del SMO

Mensajes recibidos por el SMO

Compatibilidad de configuración -solicitud-

(id fuente, nro de mensaje, id configuración hardware A, id configuración de hardware B)

Enviado por: SP

Propósito: Verificar si un par de configuraciones de hardware son compatibles o no para ser utilizadas al mismo tiempo.

Consulta de prototipos -solicitud-

(id fuente, nro de mensaje, expresión)

Contiene una consulta al SMO con el propósito de obtener los prototipos que cumplen con un determinado criterio. El criterio predica sobre los prototipos indagando sobre: subentidades, elementos, características y valores.

Enviado por: SML

Propósito: Obtener el prototipo que cumple con los requisitos que debe tener una entidad. Se utiliza para seleccionar una entidad existente del prototipo encontrado o para crear una entidad a partir del mismo.

Consulta de zona de sensado -solicitud-

(id fuente, nro de mensaje, expresión)

Contiene una consulta al SMO con el propósito de obtener las zonas de sensado que cumplen con un determinado criterio o la información de las mismas para su utilización.

Enviado por: SP

Propósito: Obtener qué zonas de sensado se podrían utilizar para sensar determinada entidad dada las características de la entidad. Un criterio podría ser “las 'zona de sensado' que tengan una determinada característica que permita un determinado valor”. También podría obtener las confiabilidades para a las distintas dimensiones.

Enviado por: SA

Propósito: Obtener las zonas de sensado necesarias para el sensado de propiocepción.

Descripción de prototipo -solicitud-

(id fuente, nro de mensaje, id prototipo, path expresion de proyección)
El propósito del mensaje es plantear una consulta al SMO para obtener valores de características, definiciones y observación de sub entidades. El parámetro id prototipo determina sobre que prototipo del SMO será aplicada la expresión del cuarto parámetro.
Enviado por: SP
Propósito: Obtener información sobre el prototipo para poder plantear la estrategia de sensado.

Distancia en característica -solicitud-

(id fuente, nro de mensaje, id característica, valor de dominio, valor de dominio)
Solicitar la evaluación en una característica la distancia relativa entre dos valores de su dominio. Esto es útil para tratar de forma homogénea a todas las características, sobre todo, cuando hay que decidir que valores serán fácilmente comparables.
Enviado por: SP
Propósito: Seleccionar según las circunstancias qué características conviene sensar.

Firma de prototipo -solicitud-

(id fuente, nro de mensaje, id prototipo, tipo de selección, orientación)
Se solicita un conjunto de tipos de características que discriminen entre todos los prototipos el prototipo pasado.
Enviado por: SP
Propósito: Construir los prejuicios para las hipótesis.

Firma de prototipo para característica -solicitud-

(id fuente, nro de mensaje, id prototipo, tipo de selección, orientación, característica)
Solicita una firma o huella de un prototipo, esto es, un conjunto de tuplas de id elemento y valor de dominio que son muy selectivos del prototipo pasado con respecto a todos los prototipos almacenados de forma que lo destaque perceptivamente para el tipo de característica pasada sobre todos los demás. Así recibe un identificador de prototipo, un tipo de selección -prejuicio positivo o negativo-, una orientación espacial y un tipo de característica.
Enviado por: SP
Propósito: Construir los prejuicios para las hipótesis.

Obtener cascara -solicitud-

(id fuente, nro de mensaje, id prototipo)
Solicitar las instrucciones para construir la superficie de contacto de una entidad.
Enviado por: SA
Propósito: Obtener los puntos de contacto de las superficies de las entidades para determinar dónde puede haber colisiones o para aplicar las fuerzas producidas por los actuadores del robot. Es usado por el motor de detección de colisiones del SA.

Obtener cuerpo -solicitud-

(id fuente, nro de mensaje, id prototipo)

Solicitar las instrucciones para construir el cuerpo mecánico de una entidad.

Enviado por: SA

Propósito: Obtener los puntos de masa de las entidades para determinar las componentes dinámicas del movimiento de las entidades. Es usado por el motor de dinámica del SA.

Renderizar -notificación-

(id fuente, nro de mensaje, prejuicio, nombre de zona sensado)

Solicitar la generación de la plantilla de comparación y la máscara de sensado de un prejuicio en una zona de sensado. A partir del prejuicio (no un id) y de la zona de sensado el SMO accede a la descripción de los prototipos del prejuicio y a la de la capacidad de sensado para construir la plantilla y la máscara en cuestión.

Enviado por: SP

Propósito: Se envía el prejuicio al SMO para que este renderice una plantilla de comparación y máscara de sensado que luego pasará el SMO directamente al SGH con el fin de comparar la salida de un sensor con la plantilla y obtener los datos de sensado a través de la máscara.

Registro de componente -solicitud-

(expresión)

Informa al SMO una definición de un componente para aplicar en su base de datos.

Enviado por: SML

Propósito: Pasar las definiciones de componentes al SMO para que todos los sistemas puedan utilizarlas.

Mensajes enviados por el SMO

Compatibilidad de configuración -respuesta-

(id fuente, nro de mensaje, valor)

Responder con un valor booleano si son o no compatibles.

Recibido por: SP

Propósito: Respuesta de la compatibilidad entre configuraciones de hardware.

Consulta de prototipos -respuesta-

(id fuente, nro de mensaje, identificadores de prototipos)

El atributo “identificadores de prototipos” es una lista con los identificadores de prototipos que cumplen con el criterio enviado. La lista puede ser vacía si no existe un prototipo que satisfaga la solicitud

Recibido por: SML

Propósito: Determinar el prototipo que cumple con una expresión de descripción de prototipo en la declaración de una entidad o de un flujo.

Consulta de zona de sensado -respuesta-

(id fuente, nro de mensaje, expresión)
Contiene el resultado de una consulta al SMO.
Recibido por: SP
Propósito: Recibir lo solicitado en la consulta para llevar a cabo la táctica de sensado.
Recibido por: SA
Propósito: Recibir lo solicitado en la consulta para obtener información de los sensores de propiocepción.

Descripción de prototipo -respuesta-

(id fuente, nro de mensaje, expresión de proyección)
Devuelve los datos seleccionados del prototipo por la path expresión de la solicitud en la expresión de proyección.
Recibido por: SP
Propósito: Indagar la constitución de los prototipos para determinar la estrategia de sensado.

Distancia en característica -respuesta-

(id fuente, nro de mensaje, valor distancia)
Devolver un valor real [0,1] que indica la distancia relativa entre los dos valores recibidos de un mismo dominio de una característica. Uno indica la mayor distancia posible y cero la menor que corresponde a cuando ambos valores pasados son idénticos.
Recibido por: SP
Propósito: Recibir cuán buena es la calidad como umbral para discriminar entre elementos cuyos valores de dominio de la característica fueron los solicitados.

Firma de prototipo -respuesta-

(id fuente, nro de mensaje, conjunto de tuplas, factor de discriminación)
Entrega un conjunto de tuplas formadas por id elemento, característica y valor de dominio. Las mismas describen elementos del prototipo solicitado que en la orientación requerida son muy selectivos del prototipo con respecto a todos los prototipos almacenados. El factor de discriminación queda reflejado como una probabilidad.
Recibido por: SP
Propósito: Obtener una descripción del prototipo que sea adecuada para ser sensada y reconocer una entidad del prototipo.

Firma de prototipo para característica -respuesta-

(id fuente, nro de mensaje, conjunto de tuplas, factor de discriminación)
Entrega un conjunto de tuplas formadas por id elemento y valor de dominio. Las mismas describen elementos del prototipo solicitado que en la orientación requerida son muy selectivos del prototipo con respecto a todos los prototipos almacenados. El factor de discriminación queda reflejado como una probabilidad.
Recibido por: SP
Propósito: Obtener la descripción del prototipo adecuada para su identificación con el sensado en base a una característica dada –por ejemplo para reutilizar un sensor ya configurado-.

Obtener cascara -respuesta-

(id fuente, nro de mensaje, seq. instrucciones)

Entregar la secuencia de instrucciones necesarias para construir la superficie de contacto de un prototipo.

Recibido por: SA

Propósito: Obtener las cáscaras de las entidades para que el motor de detección de colisiones determine los puntos de contacto entre las superficies de las entidades del contexto de acción.

Obtener cuerpo -respuesta-

(id fuente, nro de mensaje, seq. instrucciones)

Entregar la secuencia de instrucciones necesarias para construir el cuerpo dinámico de un prototipo.

Recibido por: SA

Propósito: Obtener la secuencia de instrucciones para construir el cuerpo dinámico de las entidades del contexto de acción para que el motor de dinámica pueda calcular las fuerzas intervinientes.

Registro de componente -respuesta-

(respuesta)

Informa cual fue el resultado de aplicar una definición de un componente en la base de datos del SMO.

Recibido por: SML

Propósito: Informar al SML si la definición que le suministró al SMO se registró correctamente.

Descripción de la interfaz del SGH

Mensajes recibidos por el SGH

Comando actuador -solicitud-

(id fuente, nro de mensaje, comando)

Enviado por: SA

Propósito: Enviar un comando típico de un robot diferencial de dos ruedas.

Plantilla de comparación -notificación-

(id fuente, nro de mensaje, id_prejuicio, zona de sensado, prioridad, tolerancia temporal, conjunto({nombre de dimensión, escalar de coordenada, nombre de característica, valor})

Enviado por: SMO

Propósito: Comparar la redenrización de un prejuicio con lo sensado para determinar el grado de emparejamiento.

Pedido de propiocepción -notificación-

(zona de sensado, frecuencia máxima de refresco, prioridad, frecuencia mínima de refresco)

Enviado por: SA

Propósito: Obtener un sensado rápido para las entidades propias del robot para lograr un adecuado feedback.

Máscara de obtención -notificación-

(id fuente, nro de mensaje, id_prejuicio, zona de sensado, prioridad, tolerancia temporal, conjunto(identificador de variable), conjunto({nombre de dimensión, escalar de coordenada, nombre de característica, identificador de variable}))

Enviado por: SMO

Propósito: Obtener tras el enmascaramiento producido por el renderización el valor sensado de las variables de un prejuicio.

Mensajes enviados por el SGH

Comando actuador -respuesta-

(id fuente, nro de mensaje, comando, timestamp)

Recibido por: SA

Propósito: Informar al SA si el comando se pudo llevar a cabo y el tiempo que consumió.

Emparejamiento -notificación-

(id fuente, nro de mensaje, id prejuicio, confiabilidad, error)

Recibido por: SP

Propósito: Informar al SP el grado de emparejamiento que obtuvo del sensado un prejuicio que generó.

Propiocepción directa -notificación-

(id sensor de propiocepción, conjunto(dimensión, coordenada, característica, valor))

Envía valores tomados de los sensores casi sin procesar.

Recibido por: SA

Propósito: Informar al SA la información para el feedback de las entidades propias.

Valores de variables -notificación-

(id fuente, nro de mensaje, id prejuicio, conjunto({variable, valor, confiabilidad}), error)

Recibido por: SP

Propósito: Informar los valores sensados de las variables de un prejuicio.

Descripción de la interfaz del SML

Mensajes recibidos por el SML

Arrancar recorrido -notificación-

(nombre evolución, tipo de evolución)

Este mensaje es recibido por el SML para arrancar un recorrido.

Recibe el nombre de una evolución y el tipo y el mensaje genera la instancia del recorrido.

Enviado por: SIET

Propósito: Arrancar los recorridos especulativos y descriptivos de las entidades.

Consulta de prototipos -respuesta-

(id fuente, nro de mensaje, identificadores de prototipos)

El atributo “identificadores de prototipos” es una lista con los identificadores de prototipos que cumplen con el criterio enviado. La

lista puede ser vacía si no existe un prototipo que satisfaga la solicitud.

Enviado por: SMO

Propósito: Devolver el conjunto de id de prototipos que cumplen con el criterio.

Crear entidad -respuesta-

(id fuente, nro de mensaje, id entidad)

Es el mensaje que responde el SIET con el identificador de entidad creado.

Enviado por: SIET

Propósito: Devolver el id de la entidad creada. Las confiabilidad de las dimensiones de la posición son cero y las características variables de la misma están indefinidas.

Detener recorrido -notificación-

(nombre de recorrido)

Es recibido por el SML para detener un recorrido que fue arrancado desde el exterior del SML. Lo utiliza el SIET para liberar recursos no utilizados.

Enviado por : SIET

Propósito: Cuando ya no haya más solicitudes de instantáneas para las entidades entonces el SIET detiene los recorridos especulativos y/o descriptivos correspondientes.

Entidades en modelo -respuesta-

(id fuente, nro de mensaje, identificadores de entidades)

Es un mensaje que envía el SIET con la respuesta de qué entidades satisfacen el criterio suministrado previamente por el mensaje de solicitud. El atributo “identificadores de entidades” es un conjunto de identificadores de entidad que satisfacen la solicitud previamente recibida. El conjunto puede ser vacío si no existe entidad que satisfaga la solicitud.

Enviado por: SIET

Propósito: Determinar que entidad puede aplicarse a una declaración de entidad durante la creación de un recorrido.

Instantánea -respuesta-

(id fuente, instantánea{id instantánea, id entidad, tipo, etc.})

Este mensaje es enviado por el SIET en respuesta al mensaje de solicitud. Informa la instantánea solicitada expresando su posición con respecto al contexto del solicitante –por ejemplo la entidad de referencia de un recorrido- por eso hasta que no esté registrada una entidad de referencia para la fuente este mensaje no se envía. El id de la instantánea lo determina el sistema y fuente que la crea. Por eso el mismo para poder ser globalmente referenciado tiene que considerar el sistema fuente, la fuente y un nro de secuencia dentro de la fuente.

Enviado por: SIET

Propósito: Respuesta con las instantáneas que proveen información sobre las entidades.

Instantánea fallida -notificación-

(id fuente, id instantánea, error)

Lo envía el SIET a las fuentes de instantáneas cuando no es posible llevarla a cabo.

Enviado por: SIET

Propósito: Este mensaje señala que una instantánea prescriptiva no pudo ser llevada a cabo.

Obtener característica -respuesta-

(id fuente, nro de mensaje, valor)

Envía el valor de una característica que se solicitó con el mensaje cuyo número es el enviado en el parámetro nro de mensaje.

Enviado por: SIET

Propósito: Respuesta a la solicitud de un valor de una característica de una entidad.

Operación espacio temporal -respuesta-

(id fuente, nro. de mensaje, resultado)

Este mensaje lo envía el SIET en respuesta a la solicitud respondiendo con el resultado de la evaluación del motor de álgebra espacial.

Enviado por: SIET

Propósito: Respuesta de la operación del álgebra espacio temporal.

Umbral de futuro -notificación-

(id fuente, cantidad)

Este mensaje es enviado por el SIET al SML para indicar a cada fuente cuantas instantáneas de futuro vale la pena planificar para que la probabilidad de que cambien radicalmente al llegar al presente sea baja.

Enviado por: SIET

Propósito: Este mensaje informa a los recorridos hasta cuando en el futuro es conveniente planificar.

Registro de componente -respuesta-

(respuesta)

Informa cual fue el resultado de aplicar una definición de un componente en la base de datos del SMO.

Enviado por: SMO

Propósito: Informar al SML si la definición que le suministró al SMO se registró correctamente.

Mensajes enviados por el SML

Consulta de prototipos -solicitud-

(id fuente, nro de mensaje, expresión)

Contiene una consulta al SMO con el propósito de obtener que prototipos cumplen con un determinado criterio. El criterio predica sobre los prototipos indagando sobre los elementos, las características y sus valores. El id sistema emisor es el sistema fuente.

Recibido por: SMO

Propósito: Se envía una expresión de consulta para determinar la existencia de un prototipo. Se usa antes de crear una entidad para seleccionar el prototipo que cumple con las necesidades que van planteadas en la expresión.

Crear entidad -solicitud-

(id fuente, nro de mensaje, id prototipo)

Es un mensaje recibido por el SIET y enviado por los sistemas que necesitan disponer de una entidad que no existe previamente en el modelo del SIET. Una vez que la entidad existe se pueden comenzar a realizar pedidos por las instantáneas de las mismas o enviar instantáneas prescriptivas para establecer su posición. En caso que la entidad sea virtual o abstracta la instantánea prescriptiva establece directamente su posición.

Recibido por: SIET

Propósito: Solicita crear una entidad en el SIET en base a un prototipo.

Entidades en modelo -solicitud-

(id fuente, nro de mensaje, expresión del lenguaje de consulta)

Lleva un expresión espacio temporal para evaluar en el modelo del SIET y así determinar la existencia de entidades solicitadas. El id sistema emisor es el sistema fuente.

Recibido por: SIET

Propósito: Se utiliza en la declaración de entidades para verificar si ya existen las mismas en el modelo.

Establecer característica -notificación-

(id fuente, id entidad, id característica, valor)

Es utilizado para asignar los valores de las características variables de las entidades construidas a partir de prototipos que tengas definidas características variables.

Recibido por: SIET

Propósito: Se utiliza para darle valores a las características variables de las entidades.

Instantánea -solicitud-

(id fuente, nro de mensaje, id entidad, tipo instantánea)

Este mensaje es enviado por los sistemas que consumen instantáneas al SIET. El "tipo de instantánea" puede ser "descriptiva" o "especulativa".

Recibido por: SIET

Propósito: Realizar la solicitud de necesidad de información sobre una entidad utilizada por un recorrido.

Instantánea destruida -notificación-

(id instantánea)

Es un mensaje enviado por los sistemas que generan instantáneas al SIET para informarle que destruya una instantánea.

Recibido por: SIET

Propósito: Informar al SIET que destruya una instantánea previamente notificada ya que no tiene más sentido su existencia.

Instantánea modificada - notificación-

(conjunto de restricción, instantánea{id instantánea, tipo, etc.})

Este es un mensaje enviado por los sistemas que generan instantáneas al SIET para informarle al último que actualice los datos de una instantánea previamente enviada.

Recibido por: SIET

Propósito: Modificar instantáneas ya enviadas al SIET cuando el recorrido maneja planes.

Instantánea producida -notificación-

(id fuente, conjunto de restricción, instantánea{id instantánea, id entidad, tipo, etc.})

Este mensaje es enviado al SIET para integrar la instantánea al modelo por todos los sistemas que generan cualquier tipo de instantánea. El atributo "conjunto de restricción" contiene el conjunto de identificadores de instantánea –posiblemente vacío- que regula la sincronización de la instantánea producida. El atributo "instantánea" es compuesto conteniendo todos los atributos de la instantánea enviada. El id de instantánea es creado por la fuente. Consta del id de sistema fuente, la fuente y un nro de secuencia dentro de la fuente. Así se puede garantizar la unicidad global del identificador. La instantánea debe ser del mismo tipo que el recorrido que la generó.

Recibido por: SIET

Propósito: Informar al SIET que gestione la instantánea producida por el recorrido.

Fuente activada -notificación-

(id fuente, id entidad de referencia)

Los sistemas que generan instantáneas avisan al SIET de la existencia de una nueva fuente y cuál es la entidad de referencia que usa. El identificador de entidad que se utilizará como referencia lo obtiene el sistema fuente con una consulta previa al SIET. El id sistema emisor es el sistema fuente.

Recibido por: SIET

Propósito: Informar al SIET cuál es la entidad de referencia de los recorridos para agilizar la operatoria interna del SIET.

Fuente desactivada -notificación-

(id fuente)

Es utilizado por los sistemas que generan instantáneas para avisarle al SIET que una fuente previamente informada ya no está más en funcionamiento y que por lo tanto que el SIET puede eliminar sus efectos del modelo. El id sistema emisor es el sistema fuente.

Recibido por: SIET

Propósito: Mantener acotado el tamaño del modelo del SIET para disminuir el tiempo de respuesta.

Obtener característica -solicitud-

(id fuente, nro de mensaje, id entidad, id característica, id elemento)

Este mensaje se utiliza para solicitar los valores de una característica de una entidad independientemente si la característica está definida como variable o no.

Recibido por: SIET

Propósito: Solicitar el valor de una característica de una entidad.

Operación espacio temporal -solicitud-

(id fuente, nro. de mensaje, operación)

Se envía una operación con parámetros para ser evaluada por el motor de álgebra espacial del SIET. Algunas de la funciones disponibles son :{distance(in: id instantánea, in: id instantánea): real; over(in: id instantánea, in: id instantánea): boolean; surfacedistance(in: id instantánea, in: id instantánea): real}. El id sistema emisor es el sistema fuente.

Recibido por: SIET
Propósito: Suministrar una operación del álgebra espacio temporal al SIET para que la resuelva.

Registro de componente -solicitud-

(expresión)
Informa al SMO una definición de un componente para aplicar en su base de datos.
Recibido por: SMO
Propósito: Pasar las definiciones de componentes al SMO para que todos los sistemas puedan utilizarlas.

Requerimiento -notificación-

(id fuente, id entidad, requerimiento)
Recibido por: SIET
Propósito: Registrar en el SIET los requerimientos de los recorridos para sus entidades.

Descripción de la interfaz del SP

Mensajes recibidos por el SP

Campo -respuesta-

(nro. secuencia, conjunto)
Envía un conjunto de tuplas conformadas con los atributos (identificador de entidad, id prototipo, posición con respecto a Self) Las entidades estarán dentro de la entidad virtual recibida en la solicitud -incluyendo la dimensión tiempo-. El conjunto puede estar vacío.
Enviado por: SIET
Propósito: Informar lo que para el SIET se encuentra en una región del espacio en un momento.

Compatibilidad de configuración -respuesta-

(id fuente, nro de mensaje, valor)
Responder con un valor booleano si son o no compatibles.
Enviado por: SMO
Propósito: Respuesta de la compatibilidad entre configuraciones de hardware.

Consulta de zona de sensado -respuesta-

(id fuente, nro de mensaje, expresión)
Contiene el resultado de una consulta al SMO.
Enviado por: SMO
Propósito: Recibir lo solicitado en la consulta para llevar a cabo la táctica de sensado.

Crear entidad -respuesta-

(id fuente, nro de mensaje, id entidad)
Enviado por: SIET

Propósito: Responder con el identificador de la entidad virtual creada.

Descripción de prototipo -respuesta-

(id fuente, nro de mensaje, expresión de proyección)
Devuelve los datos seleccionados del prototipo por la path expresión de la solicitud en la expresión de proyección.
Enviado por: SMO
Propósito: Indagar la constitución de los prototipos para determinar la estrategia de sensado.

Distancia en característica -respuesta-

(id fuente, nro de mensaje, valor distancia)
Devolver un valor real [0,1] que indica la distancia relativa entre los dos valores recibidos de un mismo dominio de una característica. Uno indica la mayor distancia posible y cero la menor que corresponde a cuando ambos valores pasados son idénticos.
Enviado por: SMO
Propósito: Recibir cuán buena es la calidad como umbral para discriminar entre elementos cuyos valores de dominio de la característica fueron los solicitados.

Emparejamiento -notificación-

(id fuente, nro de mensaje, id prejuicio, confiabilidad, error)
Enviado por: SGH
Propósito: Informar al SP el grado de emparejamiento que obtuvo del sensado un prejuicio que generó.

Entidades en modelo -respuesta-

(id fuente, nro de mensaje, identificadores de entidades)
Es un mensaje que envía el SIET con la respuesta de que entidades satisfacen el criterio suministrado previamente por el mensaje de solicitud. El atributo “identificadores de entidades” es un conjunto de identificadores de entidad que satisfacen la solicitud previamente recibida. El conjunto puede ser vacío si no existe entidad que satisfaga la solicitud.
Enviado por: SIET
Propósito: Obtener la capacidad de sensado actual del robot.

Instantánea -respuesta-

(id fuente, instantánea{id instantánea, id entidad, tipo, etc.})
Enviado por: SIET
Propósito: Recibir la información de las instantáneas especulativas y de las entidades virtuales que representan el alcance de las zonas de sensado activas para ser utilizadas en la confección de la estrategia de sensado del momento.

Instantánea descriptiva -solicitud-

(id fuente, id entidad, requerimiento, importancia)
Enviado por: SIET
Propósito: Iniciar el sensado de la entidad solicitada.

Instantánea fallida -notificación-

(id fuente, id instantánea, error)
Enviado por: SIET

Propósito: Informar que una instantánea prescriptiva no pudo ser llevada a cabo.

Firma de prototipo -respuesta-

(id fuente, nro de mensaje, conjunto de tuplas, factor de discriminación)

Entrega un conjunto de tuplas formadas por id elemento, característica y valor de dominio. Las mismas describen elementos del prototipo solicitado que en la orientación requerida son muy selectivos del prototipo con respecto a todos los prototipos almacenados. El factor de discriminación queda reflejado como una probabilidad.

Enviado por: SMO

Propósito: Obtener una descripción del prototipo que sea adecuada para ser sensada y reconocer una entidad del prototipo.

Firma de prototipo para característica -respuesta-

(id fuente, nro de mensaje, conjunto de tuplas, factor de discriminación)

Entrega un conjunto de tuplas formadas por id elemento y valor de dominio. Las mismas describen elementos del prototipo solicitado que en la orientación requerida son muy selectivos del prototipo con respecto a todos los prototipos almacenados. El factor de discriminación queda reflejado como una probabilidad.

Enviado por: SMO

Propósito: Obtener la descripción del prototipo adecuada para su identificación con el sensado en base a una característica dada –por ejemplo para reutilizar un sensor ya configurado-.

Obtener característica -respuesta-

(id fuente, nro de mensaje, valor)

Enviado por: SIET

Propósito: Obtener el valor de una característica de una entidad.

Operación espacio temporal -respuesta-

(id fuente, nro. de mensaje, resolución)

Enviado por: SIET

Propósito: Respuesta de la evaluación si las entidades virtuales espacios incógnitos están o no contenidas en las las entidades virtuales que representan el alcance de las zonas de sensado.

Valores de variables -notificación-

(id fuente, nro de mensaje, id prejuicio, conjunto({variable, valor, confiabilidad}), error)

Enviado por: SGH

Propósito: Informar los valores sensados de las variables de un prejuicio.

Mensajes enviados por el SP

Campo -solicitud-

(id fuente, nro. de mensaje, id entidad virtual)

Lo envía el SP para obtener desde el SIET qué entidades están presentes alrededor de una hipótesis. Precisamente la entidad virtual determina el espacio dónde se realizará la búsqueda.

Recibido por: SIET

Propósito: Solicitar al SIET qué entidades existen y en qué posiciones en una determina región del espacio.

Compatibilidad de configuración -solicitud-

(id fuente, nro de mensaje, id configuración hardware A, id configuración de hardware B)

El SMO recibe un par de configuraciones de hardware y evalúa si son compatibles o no para ser utilizadas al mismo tiempo.

Recibido por: SMO

Propósito: Verificar si las configuraciones de hardware tomadas de los prejuicios son compatibles de ser usadas al mismo tiempo.

Consulta de zona de sensado -solicitud-

(id fuente, nro de mensaje, expresión)

Contiene una consulta al SMO con el propósito de obtener las zonas de sensado que cumplen con un determinado criterio o la información de las mismas para su utilización.

Recibido por: SMO

Propósito: Obtener qué zonas de sensado se podrían utilizar para sensar determinada entidad dada las características de la entidad. Un criterio podría ser “las 'zona de sensado' que tengan una determinada característica que permita un determinado valor”. También podría obtener las confiabilidades para a las distintas dimensiones.

Crear entidad -solicitud-

(id fuente, nro de mensaje, id prototipo)

Este mensaje es enviado por los sistemas que necesitan disponer de una entidad que no existe previamente en el modelo del SIET. Una vez que la entidad existe se pueden comenzar a realizar pedidos por las instantáneas de las mismas o enviar instantáneas prescriptivas para establecer su posición.

Recibido por: SIET

Propósito: que el SP pueda crear entidades virtuales en el modelo para usar el SIET como una memoria espacio temporal para llevar a cabo el sensado.

Descripción de prototipo -solicitud-

(id fuente, nro de mensaje, id prototipo, path expresion de proyección)

El propósito del mensaje es plantear una consulta al SMO para obtener valores de características, definiciones y observación de sub entidades. El parámetro id prototipo determina sobre que prototipo del SMO será aplicada la expresión del cuarto parámetro.

Recibido por: SMO

Propósito: Obtener información sobre el prototipo para poder plantear la estrategia de sensado.

Distancia en característica -solicitud-

(id fuente, nro de mensaje, id característica, valor de dominio, valor de dominio)

Solicitar la evaluación en una característica la distancia relativa entre dos valores de su dominio. Esto es útil para tratar de forma homogénea a todas las características, sobre todo, cuando hay que decidir que valores serán fácilmente comparables.

Recibido por: SMO

Propósito: Seleccionar según las circunstancias qué características conviene sensar.

Entidades en modelo -solicitud-

(id fuente, nro de mensaje, expresión del lenguaje de consulta)

Recibido por: SIET

Propósito: Obtener las entidades virtuales de tipo *range* que están activas en cada momento junto con las entidades sensor a la cuales están asociadas.

Establecer característica -notificación-

(id fuente, id entidad, id característica, valor)

Recibido por: SIET

Propósito: Se utiliza para almacenar las características variables de las entidades.

Instantánea -solicitud-

(id fuente, nro de mensaje, id entidad, tipo instantánea)

Este mensaje es enviado por los sistemas que consumen instantáneas del SIET. El "tipo de instantánea" puede ser "descriptiva" o "especulativa".

Recibido por: SIET

Propósito: Solicitar instantáneas especulativas para obtener la información del dominio de aplicación del robot, de la posición de las entidades del alcance de los sensores y de las entidades espacio incógnito denotadas en los prejuicios.

Instantánea descriptiva -respuesta-

(id entidad, instantánea{id instantánea, tipo, etc.})

Recibido por: SIET

Propósito: Integrar en el modelo las instantáneas descriptivas presentes de las entidades que esté sensando el SP en respuesta a los requerimientos del SIET.

Instantánea destruida -notificación-

(id instantánea)

Recibido por: SIET

Propósito: Eliminar del modelo una instantánea prescriptiva del futuro previamente notificada por el SP ya sea porque aquel futuro ya no es posible o porque ya no es más necesitado.

Instantánea modificada -notificación-

(conjunto de restricción, instantánea{id instantánea, tipo, etc.}[, configuración de hardware])

Recibido por: SIET

Propósito: Ajustar el plan de movimiento del sensado activo de un modo más eficiente que destruyendo y creando instantáneas muy similares.

Instantánea producida -notificación-

(id fuente, conjunto de restricción, instantánea{id instantánea, id entidad, tipo, etc.}[, configuración de hardware])

El atributo "conjunto de restricción" contiene el conjunto de identificadores de instantáneas –posiblemente vacío- que regula la sincronización de la instantánea producida. El atributo "instantánea" es compuesto conteniendo todos los atributos de la instantánea enviada. El id de instantánea es creado por la fuente. Consta del id de sistema fuente, del id de la fuente y un nro de secuencia dentro de la fuente. Así se puede garantizar la unicidad global del identificador.

Recibido por: SIET

Propósito: Integrar en el modelo el movimiento de las entidades zona de sensado para el sensado activo y registrar las dimensiones de las entidades virtuales usadas por el SP como memorias espacio temporales.

Firma de prototipo -solicitud-

(id fuente, nro de mensaje, id prototipo, tipo de selección, orientación)

Se solicita un conjunto de tipos de características que discriminen entre todos los prototipos el prototipo pasado.

Recibido por: SMO

Propósito: Construir los prejuicios para las hipótesis.

Firma de prototipo para característica -solicitud-

(id fuente, nro de mensaje, id prototipo, tipo de selección, orientación, característica)

Solicita una firma o huella de un prototipo, esto es, un conjunto de tuplas de id elemento y valor de dominio que son muy selectivos del prototipo pasado con respecto a todos los prototipos almacenados de forma que lo destaque perceptivamente para el tipo de característica pasada sobre todos los demás. Así recibe un identificador de prototipo, un tipo o de selección -prejuicio positivo o negativo-, una orientación espacial y un tipo de característica.

Recibido por: SMO

Propósito: Construir los prejuicios para las hipótesis.

Fuente activada -notificación-

(id fuente, id entidad de referencia)

Notifica al SIET de una nueva fuente y la entidad de referencia que la misma utiliza.

Recibido por: SIET

Propósito: Registrar en el SIET las fuentes coadyuvantes.

Fuente desactivada -notificación-

(id fuente)

Recibido por: SIET

Propósito: Registrar en el SIET las fuentes coadyuvantes que han sido desactivadas.

Liberar entidad -notificación-

(id fuente, id entidad)

Recibido por: SIET
Propósito: Liberar el uso de las entidades que no son más necesarias para los procesos del SP generalmente entidades virtuales.

Obtener característica -solicitud-

(id fuente, nro de mensaje, id entidad, id característica, id elemento)
Este mensaje se utiliza para solicitar los valores de una característica de una entidad independientemente si la característica está definida como variable o no.
Recibido por: SIET
Propósito: Solicitar información de las entidades del modelo del SIET.

Operación espacio temporal -solicitud-

(id fuente, nro. de mensaje, operación)
Se envía una operación con parámetros para ser evaluada por el motor de álgebra espacial del SIET. Algunas de las operaciones disponibles son:

- distance (in: id instantánea, in: id instantánea): real
- over (in: id instantánea, in: id instantánea): boolean
- surfacedistance (in: id instantánea, in: id instantánea): real

"operación" es de un tipo de datos especial que lleva en forma eficiente la invocación de la operación para que no tenga que ser parseada en el momento de la resolución. El id sistema emisor es el sistema fuente.
Recibido por: SIET
Propósito: Evaluar si la entidades objetivo de los prejuicios están contenidas en las zonas de sensado.

Renderizar -notificación-

(id fuente, nro de mensaje, prejuicio, nombre de zona sensado)
Solicitar la generación de la plantilla de comparación y la máscara de sensado de un prejuicio en una zona de sensado. A partir del prejuicio (no un id) y de la zona de sensado el SMO accede a la descripción de los prototipos del prejuicio y a la de la capacidad de sensado para construir la plantilla y la máscara en cuestión.
Recibido por: SMO
Propósito: Se envía el prejuicio al SMO para que este renderice una plantilla de comparación y máscara de sensado que luego pasará el SMO directamente al SGH con el fin de comparar la salida de un sensor con la plantilla y obtener los datos de sensado a través de la máscara.

Descripción de la interfaz del SA

Mensajes recibidos por el SA

Comando actuador -respuesta-

(id entidad actuador, nro. de secuencia de comando, respuesta, timestamp**)**
Mensaje proveniente del SGH hacia el SA que indica si el comando fue llevado a cabo o no. El timestamp se utiliza para computar el índice de ejecución.
Enviado por: SGH

Propósito: informar si el comando se pudo llevar a cabo y en caso afirmativo indicar cuando se hizo.

Consulta de zona de sensado -respuesta-

(id fuente, nro de mensaje, expresión)
Contiene el resultado de una consulta al SMO.
Enviado por: SMO
Propósito: Recibir lo solicitado en la consulta para obtener información de los sensores de propiocepción.

Entidad dentro de contexto de acción -notificación-

(id entidad)
Informa cuando una entidad física entra dentro del contexto de acción.
Enviado por: SIET
Propósito: Informar al motor de dinámica y de detección de colisiones las entidades físicas que debe considerar en sus modelos internos.

Entidad fuera de contexto de acción -notificación-

(id entidad)
Informa cuando una entidad física sale el contexto de acción.
Enviado por: SIET
Propósito: Informar al motor de dinámica y de detección de colisiones las entidades físicas que debe considerar en sus modelos internos.

Instantánea -respuesta-

(id fuente, instantánea{id instantánea, id entidad, tipo, etc.})
Este mensaje es enviado por el SIET en respuesta al mensaje de solicitud. Informa la instantánea solicitada expresando su posición con respecto al contexto del solicitante -por ejemplo Self en este caso- por eso hasta que no esté registrada una entidad de referencia para la fuente este mensaje no se envía. El id de la instantánea lo determina el sistema y fuente que la crea. Por eso el mismo para poder ser globalmente referenciado tiene que considerar el sistema fuente, la fuente y un nro de secuencia dentro de la fuente.
Enviado por: SIET
Propósito: Responder con las instantáneas que proveen información sobre las entidades solicitadas.

Instantánea declarativa -notificación-

(conjunto de restricción, instantánea {id instantánea, tipo, etc.})
El mensaje es enviado por el SIET al SA para informarle la instantánea -sea de una entidad propia o no- para materializar. El atributo “conjunto de restricción” contiene el conjunto de identificadores de instantánea –posiblemente vacío- que regula la sincronización de la instantánea seleccionada y el atributo “instantánea” es compuesto conteniendo todos los atributos de dicha instantánea. Este mensaje puede enviarse más de una vez para una misma instantánea lo cual indica una actualización de los datos asociados a esta.
Enviado por: SIET
Propósito: Recibir las instantáneas que debe materializar.

Obtener cascara -respuesta-

(id fuente, nro de mensaje, seq. instrucciones)

Entregar la secuencia de instrucciones necesarias para construir la superficie de contacto de un prototipo.

Enviado por: SMO

Propósito: Obtener las cáscaras de las entidades para que el motor de detección de colisiones determine los puntos de contacto entre las superficies de las entidades del contexto de acción.

Obtener cuerpo -respuesta-

(id fuente, nro de mensaje, seq. instrucciones)

Entregar la secuencia de instrucciones necesarias para construir el cuerpo dinámico de un prototipo.

Enviado por: SMO

Propósito: Obtener la secuencia de instrucciones para construir el cuerpo dinámico de las entidades del contexto de acción para que el motor de dinámica pueda calcular las fuerzas intervinientes.

Propiocepción directa -notificación-

(id sensor de propiocepción, conjunto(dimensión, coordenada, característica, valor))

Envía valores tomados de los sensores casi sin procesar.

Enviado por: SGH

Propósito: Obtener el feedback de los sensores de las entidades propias suficientemente rápido como para adecuarse al tiempo real.

Tamaño de muestra para índice de ejecución -notificación-

(id fuente, cantidad, período de notificación)

El parámetro *cantidad* informa la cantidad de instantáneas prescriptivas a considerar y el *período de notificación* cada cuanto tiene que interrumpir el SA para informar el resultado.

Enviado por: SIET

Propósito: Obtener los parámetros utilizados para calcular índices de performance de la actuación.

Mensajes enviados por el SA

Comando actuador -solicitud-

(id fuente, nro de mensaje, comando)

Enviar un comando de una secuencia de comando.

Recibido por: SGH

Propósito: Enviar un comando típico de un robot diferencial de dos ruedas.

Consulta de zona de sensado -solicitud-

(id fuente, nro de mensaje, expresión)

Contiene una consulta al SMO con el propósito de obtener las zonas de sensado que cumplen con un determinado criterio o la información de las mismas para su utilización.

Recibido por: SMO

Propósito: Obtener las zonas de sensado necesarias para el sensado de propiocepción.

Desarrollo de aplicación fallido -notificación-

(id instantánea, error)

Recibido por: SIET

Propósito: notificar al SIET cuando el SA no se puede generar un desarrollo de instantáneas prescriptivas procedimentales que satisfaga lo necesario de una instantánea prescriptiva seleccionada por el SIET.

Instantánea -solicitud-

(id fuente, nro de mensaje, id entidad, tipo instantánea)

Este mensaje es enviado por los sistemas que consumen instantáneas del SIET. El "tipo de instantánea" puede ser "descriptiva" o "especulativa".

Recibido por: SIET

Propósito: Obtener instantáneas descriptivas de las entidades del contexto de acción y de las entidades propias con el fin de recalibrar los sensores de propiocepción.

Instantánea destruida -notificación-

(id instantánea)

Recibido por: SIET

Propósito: Eliminar del modelo una instantánea prescriptiva propia procedimental previamente notificada por el SA dado que ya no es posible operar sobre la entidad recurso en cuestión o porque se asignó el recurso a otro fin.

Instantánea modificada -notificación-

(conjunto de restricción, instantánea{id instantánea, tipo, etc.}[, configuración de hardware])

Recibido por: SIET

Propósito: Actualizar el plan de movimiento de entidades propias de tipo actuador de un modo más eficiente que destruyendo y creando instantáneas muy similares.

Instantánea producida -notificación-

(id fuente, conjunto de restricción, instantánea{id instantánea, id entidad, tipo, etc.}[, configuración de hardware])

El atributo “conjunto de restricción” contiene el conjunto de identificadores de instantáneas –posiblemente vacío- que regula la sincronización de la instantánea producida. El atributo “instantánea” es compuesto conteniendo todos los atributos de la instantánea enviada. El id de instantánea es creado por la fuente. Consta del id de sistema fuente, del id de la fuente y un nro de secuencia dentro de la fuente. Así se puede garantizar la unicidad global del identificador.

Recibido por: SIET

Propósito: Integrar en el SIET las instantáneas prescriptivas de entidades propias de tipo actuador procedimentales generadas a partir de las instantáneas prescriptivas declarativas.

Fuente activada -notificación-

(id fuente, id entidad de referencia)

Notifica al SIET de una nueva fuente y la entidad de referencia que la misma utiliza.

Recibido por: SIET

Propósito: Registrar en el SIET las fuentes coadyuvantes.

Fuente desactivada -notificación-

(id fuente)

Recibido por: SIET
Propósito: Registrar en el SIET las fuentes coadyuvantes que han sido desactivadas.

Índice de ejecución -notificación-

(id fuente, valor)
Recibido por: SIET
Propósito: Notificar al SIET los valores del índice de ejecución para cada fuente.

Materialización fallida -notificación-

(id instantánea, error)
Recibido por: SIET
Propósito: Informar al SIET los errores producidos por problemas con los recursos; ya sea por falla o por disponibilidad.

Obtener cascara -solicitud-

(id fuente, nro de mensaje, id prototipo)
Solicitar las instrucciones para construir la superficie de contacto de una entidad.
Recibido por: SMO
Propósito: Obtener los puntos de contacto de las superficies de las entidades para determinar dónde puede haber colisiones o para aplicar las fuerzas producidas por los actuadores del robot. Es usado por el motor de detección de colisiones.

Obtener cuerpo -solicitud-

(id fuente, nro de mensaje, id prototipo)
Solicitar las instrucciones para construir el cuerpo mecánico de una entidad.
Recibido por: SMO
Propósito: Obtener los puntos de masa de las entidades para determinar las componentes dinámicas del movimiento de las entidades. Es usado por el motor de dinámica.

Pedido de propiocepción -notificación-

(zona de sensado, frecuencia máxima de refresco, prioridad, frecuencia mínima de refresco)
Recibido por: SMO
Propósito: Obtener un sensado rápido para las entidades propias del robot para lograr un adecuado feedback.

Requerimiento -notificación-

(id fuente, id entidad, requerimiento)
Recibido por: SIET
Propósito: Registrar en el SIET los requerimientos de las fuentes coadyuvantes para sus entidades.

Estructura de los requerimientos

Requerimiento especulativo

Atributos comunes para entidades concretas, virtuales y abstractas

entity.variation	cantidad de variaciones a generar para la entidad.
entity.actualizaciontime	Frecuencia de actualización para la dimensión *time*.
entity.actualizacionproccestime	Frecuencia de actualización para la dimensión *procestime*.

Atributos para entidades concretas y virtuales

distance.resolution	requerimiento de resolución.
distance.confiability	requerimiento de probabilidad de que el valor procesado esté en el valor de la distancia más menos la tolerancia.
Course	análogo a distance.
Height	análogo a distance.
Yaw	análogo a distance.
Roll	análogo a distance.
Pitch	análogo a distance.

Requerimiento descriptivo

Atributos para entidades concretas y virtuales

prototype.confiability	confiabilidad necesitada en la identificación del prototipo de la entidad.
entity.confiability	confiabilidad necesitada en la identificación de la entidad.
distance.resolution	requerimiento de resolución.
distance.confiability	requerimiento de probabilidad de que el valor materializado esté en el valor de la distancia más menos la tolerancia.
course	análogo a distance.
height	análogo a distance.
yaw	análogo a distance.
roll	análogo a distance.
pitch	análogo a distance.

Requerimiento prescriptivo

Atributos para entidades concretas y virtuales

distance.tolerance	delta en que el valor de la distancia puede ser materializado.
distance.confiability	requerimiento de probabilidad de que el valor materializado esté en el valor de la distancia más menos la tolerancia.
course	análogo a distance.
height	análogo a distance.
yaw	análogo a distance.
roll	análogo a distance.
pitch	análogo a distance.

Estructura de las instantáneas

Instantánea especulativa

Atributos comunes para entidades concretas, virtuales y abstractas

entity.variation	identificador de la variación.
entity.time	valor de la dimensión *time*.
entity.hitointerrupcion	indica si es un hito o no de interrupción.
entity.processtime	valor del atributo *processtime*.
Entity.main	Valor que indica si es la variación principal o no.
identity	Identificación de la entidad.
Type	Siempre especulativa en este caso.

Atributos para entidades concretas y virtuales

distance	valor en la dimensión distance de la entidad.
distance.tolerance	delta en que el valor real de la distancia puede variar.
distance.confiability	confiabilidad en que el valor real de la distancia de la entidad este dentro de la tolerancia.
Course	análogo a distance.
Height	análogo a distance.
Yaw	análogo a distance.
Roll	análogo a distance.
Pitch	análogo a distance.

Instantánea descriptiva

Atributos para entidades concretas y virtuales

prototype.confiability	confiabilidad en la identificación del prototipo.
entity.confiability	confiabilidad en la identificación de la entidad.
distance	valor de la dimensión distance de la entidad.
distance.tolerance	delta en que el valor real de la distancia puede variar.
distance.confiability	confiabilidad que en la distancia de la entidad este dentro de la tolerancia.
course	análogo a distance.
height	análogo a distance.
yaw	análogo a distance.
roll	análogo a distance.
pitch	análogo a distance.
time	análogo a distance.
entity.processtime	valor del atributo *processtime*.
identity	Identificador de la entidad.
Type	Siempre descriptiva en este caso.

Instantánea prescriptiva

Atributos para entidades concretas y virtuales

identity	Identificación de la entidad.
distance	valor en la dimensión distance de la entidad.
course	análogo a distance.
height	análogo a distance.
yaw	análogo a distance.
roll	análogo a distance.
pitch	análogo a distance.
time	análogo a distance.

processtime	valor del atributo *processtime*.
mode	Declarativo o procedimental
Type	Siempre prescriptiva en este caso.

Gramática del lenguaje

A continuación se muestra la gramática del lenguaje. En itálica y alineado a la derecha se agrega cuando es necesario una aclaración para reflejar los aspectos que dependen del contexto. El no terminal inicial es "**<inicial>".** En negrita se presentan los terminales del lenguaje. Los no terminales están entre paréntesis angulares. Terminales de un carácter se presentan entre comillas simples. Los símbolos de la gramática son "::=", "|", "'", "'", ".", "(", ")", "<", ">", "[", "]".

<inicial> ::= <lista de declaraciones> <lista de evoluciones>.

<lista de declaraciones> ::= <declaración> <lista de declaraciones> | <declaración>.

<declaración> ::=
<declaración de característica> | <declaración de elemento> | <declaración de prototipo>.

<declaración de característica> ::=
caracteristic ID (**physical** | **abstract**)
<tipo elemental> <unidad de referencia> [<dominio>] '**;**'
end.

<tipo elemental> ::= **integer** | **float** | **double** | **enum** | **boolean** | **time** | **date** | **datetime** .

<unidad de referencia> ::= **gram** | **second** | **mm** | **hz** | **undimensional**.
La unidad de referencia está integrada al lenguaje así como sus conversiones y escalas. Las variables por lo tanto no sólo tiene un tipo de datos que los represente sino también una unidad física. La presente lista es muy limitada.

<dominio> ::= ('[' | '**(**') <número> '**,**' <número> ('**)**' | '**]**') | '[' <enumeración> '**]**'.

<enumeración> ::= **ID** '**,**' <enumeración> | **ID** .

<declaración de elemento> ::=
element ID (**concrete** | **virtual**)
importNURBS '(' datosNURBS ')'
<lista de características>
Los elementos concretos deben tener al menos una característica física La diferencia entre prototipos y elementos que los elementos no se pueden referenciar en las evoluciones y que los prototipos tienen mayor poder de expresividad para describir su constitución .
end .

<lista de características> ::= <característica> <lista de características> | .

<característica> ::= **ID** (**fix** <valor característica> |
constant [**default** <valor característica>] |
variable [**default** <valor característica>]) '**;**' .
ID corresponde al nombre de una declaración de característica.

<declaración de prototipo> ::=
prototype (**ID** | **self**) (**concrete** | **virtual** | **abstract**)
Es obligatorio que el prototipo self esté declarado y self siempre debe ser concreto.
<lista de declaración de partes privadas>
construction
<lista de instancias>
Un prototipo concreto puede tener instancias de cualquier tipo pero debe tener al menos una concreta. Un prototipo virtual sólo puede tener instancias virtuales y abstractas pero debe tener al menos una virtual. Un prototipo abstracto sólo puede tener instancias abstractas.
<lista de características>
end
[
Binding sólo es válida para el prototipo self.
binding
<lista de declaración de zona de sensado>
end
]
end.

<lista de declaración de partes privadas> ::=
(<declaración de prototipo> |
<declaración de elemento>) <lista de declaración de partes privadas> | .

<lista de declaración de zona de sensado> ::=
<declaración de zona de sensado> <lista de declaración de zona de sensado> |
<declaración de zona de sensado> .

<declaración de zona de sensado> ::=
sensingzone
Las sensingzone sólo son válidas para el prototipo self.
sensor ID '**;**'
Debe ser el id de una entidad sensor.
hardwareconfiguration ID '**;**'
Este ID de configuración de hardware se define aquí que es para la entidad sensor anterior.
caracteristic ID '**;**'
ID es el nombre de una característica que es sensable por la entidad sensor.
range ID '**;**'
El ID de range es el de una subentidad virtual del sensor de la zona de sensado que describe el alcance del sensor para la característica con la configuración de hardware. Por ser una subentidad de tipo range la posición de la misma es con respecto a la entidad sensor.
resolution <lista de resolución> '**;**'
end.

<lista de instancias> ::= <instancia> <lista de instancias> | .

<instancia> ::=
instance [**ID**] **for** (**prototype ID** | **element ID**)
is (sensor | actuator | pasive | <u>foreign</u> | range)
Las instancias foreign son el default.
<lista de asignación de características>
[<restricción espacial>]

La restricción espacial es sólo obligatoria en la entidades y elementos concretos. Opcional en los virtuales y no es válida para las abstractas.

end.

<lista de asignación de características> ::=
<asignación de característica> <lista de asignación de características> | .

<asignación de característica> ::= **ID** '=' <valor característica> '**;**' .
ID corresponde al nombre de una característica que este definida como constante o variable.

<posición espacial> ::=
position
course <número><unidad de ángulo> '**;**'
distance <número><unidad de longitud> '**;**'
height <número><unidad de ángulo> '**;**'
pitch <número><unidad de ángulo> '**;**'
roll <número><unidad de ángulo> '**;**'
yaw <número><unidad de ángulo> '**;**'
end.

<restricción espacial>[20] ::=
spatial restriction
[**course** <restricción angular> '**;**']
[**distance** <restricción longitud> '**;**']
[**height** <restricción angular> '**;**']
[**pitch** <restricción angular> '**;**']
[**roll** <restricción angular> '**;**']
[**yaw** <restricción angular> '**;**']
end.

<restricción angular> ::= <número><unidad de ángulo> |
(**between** <número><unidad de ángulo> **and** <número><unidad de ángulo> | **unlimit**)
[**default** <número><unidad de ángulo>].

<restricción longitud> ::= <número><unidad de longitud> |
between <número><unidad de longitud> **and** <número><unidad de longitud>
[**default** <número><unidad de longitud>].

<lista de resolución> ::= <elemento de resolución> <lista de resolución>
| <elemento de resolución> .

<elemento de resolución> ::= '**(**'
(**course |**
distance |
height |
pitch |
roll |
yaw
) '**,**' '**{**' <lista de clase> '**}**'
'**)**'.

<lista de clase> ::= <lista de clase> <clase> | <clase>

[20]La restricción espacial no puede expresar directamente todas las restricciones posibles ya que se limita a las "ejes" de las dimensiones.

<clase> ::= '**(**'
<número>
<número> es una probabilidad que indica la confiabilidad de los valores de dominio en la dimensión.
'**,**' '**{**' <lista valores dominio> '**}**'
'**)**'.

<lista valores dominio> ::= <lista valores dominio> <elemento de valores de dominio>
| <elemento de valores de dominio> **.**

<elemento de valores de dominio> ::= ('**{**' <lista valores dominio extensiva> '**}**'
| '**(**' <valor dominio> '**,**' <valor dominio> '**)**')

<lista valores dominio extensiva> ::= <lista valores dominio extensiva>
'**,**' <valor característica>
| <valor característica> .

<lista de evoluciones> ::=
<declaración de evolución> <lista de evoluciones> | <declaración de evolución>.

<declaración de evolución> ::=
evolution (**descriptive** | **prescriptive** | **especulative**) **ID**
[
state
<lista de declaración de variable>
end state
]
context around ID
El ID es el de una entidad declarada en context.
<lista de elementos de context>
end context
[
requirement
<lista de rasgos>
<lista de estilos>
end requirement
]

produce
on
reaction velocity <frecuencia>
planification velocity <frecuencia>
horizon<periodo>
delta time <porcentaje>
delta processtime <porcentaje>
for

[
start
<lista sentencias>
end start
]
[
while
[
switch milestone
<lista de casos>
end switch milestone

]
[
rhythm
<lista de paso de ritmo>
end rhythm
]
[
perform on <lista de entidades>
Los ID de entidades deben estar definidos en el contexto.
do
<lista sentencias>
end perform
]
end while
]

[
finish
<lista sentencias>
end finish
]
[
resource
<lista sentencias>
end resource
]
end produce
[
syncronization
<lista de abouts>
<lista de transiciones>
end syncronization
]
end evolution .

<lista de declaración de variable> ::=
<declaración variable> <lista de declaración de variable>
| <declaración variable> .

<declaración variable> ::= <tipo elemental> <unidad de referencia> [<dominio>] **ID** '**;**'.

<lista de elementos de context> ::=
<elemento de context> <lista de elementos de context> | <elemento de context>.

<elemento de context> ::=
(
travel ID '**:**' **ID**
El primer ID es un nombre de evolución, el segundo el identificador de recorrido que se le asigna localmente.
|
(**starring | cast | guest)**
[ID] '**.**' '**(**' <expresión de entidad> '**)**' '**:**' **ID ['[' ']']**
El primer ID es un nombre de un prototipo, el segundo es el identificador que se le asigna localmente a la entidad.
). '**;**'.

<lista de rasgos> ::= <rasgo> <lista de rasgos> | <rasgo> .

<rasgo> ::=
feature (**descriptive** | **prescriptive** | **especulative**) **ID** '**:**' **ID is**
El primer ID es el de una entidad declarada en el contexto, el segundo el identificador asignado al rasgo.
<lista de requerimientos>
end feature .

<lista de requerimientos> ::= <requerimiento> <lista de requerimientos> | <requerimiento> .

<requerimiento> ::= (
entity '.' **(** **confiability** '**:**' <número> **confidence** |
variation ':' <número> **undimensional**
) |
(**distance** | **course** | **height** | **yaw** | **roll** | **pitch**) '.' **confiability** ':' <número> **confidence** |
distance '.' (**tolerance** | **resolution**) ':' <número> <unidad de longitud>
(**course** | **height** | **yaw** | **roll** | **pitch**) '.'
(**tolerance** | **resolution**) ':' <número> <unidad de ángulo>
) ';'.

<lista de estilos> ::= <estilo> <lista de estilos> | <estilo>.

<estilo> ::=
style ID (**primer** |)
Uno y sólo un estilo tiene que estar calificado como primer
<lista de identificadores de rasgos>
end style .

<lista de identificadores de rasgos> ::=
ID '**;**' <lista de identificadores de rasgos> | **ID** '**;**' .
Los IDs corresponden a rasgos definidos en requeriment.

<lista de casos> ::= <caso> <lista de casos> | <caso>.

<caso> ::=
case '(' <condición> **')'**
<lista sentencias>
end.

<lista de paso de ritmo> ::= <paso de ritmo> <lista de paso de ritmo> | <paso de ritmo>.

<paso de ritmo> ::= **ID** <período> '**;**' .
El ID corresponde a un estilo definido en requeriments.

<lista de abouts> ::= <about> <lista de abouts> | <about>.

<about> ::=
about ID
El ID corresponde a un recorrido definido en el contexto.
<lista de triggers>
end about .

<lista de triggers> ::= <trigger> <lista de triggers> | <trigger> .

<trigger> ::=
trigger ID
situation around ID '[' <condición> **']'**
ID es una entidad definida en context que es utilizada como de referencia.

when (**first-time** | **each_time** | **last_time**)
fire ID

El ID corresponde a una transición que tiene que estar definida en la sección syncronization.

end trigger .

<lista de transiciones> ::= <transición> <lista de transiciones> | <transición>.

<transición> ::=
transition ID
over '**(**' <lista de nombres de triggers> '**)**'
<expresión de sincronización>
end transition .

<lista de nombres de triggers> ::= <nombre de trigger> '**,**' <lista de nombres de triggers>
| <nombre de trigger> .

<nombre de trigger> ::= **ID** '**.**' **ID.**

El primer ID del par ID.ID se corresponde a un recorrido declarado en un about y el segundo con el nombre del trigger perteneciente al mismo about.

<expresión de sincronización> ::=
<nombre de trigger> |
arrange '**(**' <expresión de sincronización> '**,**' <expresión de sincronización> '**)**' |
inhibit '(' <expresión de sincronización> '**)**' |
abort '**(**' <expresión de sincronización> '**)**' |
conclude '**(**' <expresión de sincronización> '**)**' |
lever '**(**' <llamada de recorrido> '**)**'.

<lista sentencias> ::= <lista sentencias> <sentencia> | .

<sentencia> ::= <sentencia compuesta> |
<sentencia condicional> |
<sentencia de asignación> |
changeStyle ID '**;**' |
setInterrupt ID ';' |
raise ';' |
retry ';' |
<expresion de recorrido> '**;**' .

<sentencia compuesta> ::= '{' <lista de declaración de variable> <lista sentencias> '}'.

<sentencia condicional> ::=
if '**(**' <condición> '**)**'
<lista sentencias>
else
<lista sentencias>
end.

<sentencia de asignación> ::= <lista de nombres con puntos> '=' <expresión> '**;**' .

<expresión de recorrido> ::= **ID** '**(**' <llamada de recorrido> '**)**'.

<llamada de recorrido> :: = **command** '**:**' (start | stop) <lista de entidades invitadas>.

<lista de entidades invitadas> ::= <entidad invitada> <lista de entidades invitadas> | .

<entidad invitada> ::= **ID** ‘**:**’ **ID**

El primer ID es el nombre local formal de una invitada definida en la evolución. El segundo ID es el nombre de la entidad pasada en el contexto del recorrido llamador.

<condición> ::=
<condición> **and** <condición> |
<condición> **or** <condición> |
not <condición> |
(<condición> **)** |
<expresión> <op_rel> <expresión> .

<expresión> ::= <expresión> <op_binario> <expresión> |
<op_unario> <expresión> |
<lista de nombres con puntos> |
<constante> |
<función> .

<op_rel> ::= ‘**<=**’ | ‘**<**’ | ‘**>**’ | ‘**>=**’ | ‘**==**’ | ‘**!=**’ .

<op_binario> ::= ‘**+**’ | ‘**-**’ | ‘***’ | ‘/’ | ‘^’ .

<op_unario> ::= ‘**-**’ .

<lista de nombres con puntos> ::= **ID** ‘.’ <lista de nombres con puntos> | **ID** .

<función> ::= **ID** ‘**(**’ <lista de expresiones con coma> ‘**)**’.

<lista de expresiones con coma> ::= <expresión> ‘**,**’ <lista de expresiones con coma>
| <expresión> .

<constante> ::= **CONST**.

<expresión de entidad> ::= (**where** <clausula_where> |
new |
ID |
choose) .

<clausula_where> ::= TBD.

Debe ser definido. Fuera de alcance del trabajo.

<período> ::= <número> <unidad de tiempo> .
<frecuencia> :: = <número> **hz** .
<porcentaje> ::= <número> **‘%’** .

<número> ::= NUM_REAL .
<unidad de ángulo> ::= **grade**.
<unidad de longitud> ::= (**mm** | **cm** | **dm** | **m**).
<unidad de tiempo> ::= (**ms** | **cs** | **ds** | **s** | **min**).
<valor característica> ::= **ID**.

Es un valor perteneciente el dominio de la característica.

BIBLIOGRAFÍA

Libros

- [1] Arkin, R.C., Behavior-Based Robotics. MIT Press, MA, 1998.
- [2] Borenstein, J., Everett, H.R., Feng, L., Where Am I? Sensors and Methods for Mobile Robot Positioning. Ann Arbor, University of Michigan. Available at http:// www-personal.engin.umich.edu/~johannb/position.htm.
- [3] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S., Principles of Robot Motion Theory, Algorithms, and Implementation, Cambridge MIT Press, MA, 2005.
- [4] Florczyk, S., Robot Vision: Video-based Indoor Exploration with Autonomous and Mobile Robots, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim , 2005.
- [5] Sandin, P. E., Robot Mechanisms and Mechanical Devices Illustrated , McGraw-Hill , New York , 2003.
- [6] Siegwart, R., Nourbakhsh I., Introduction to Autonomous Mobile Robots , MIT Press, MA, 2004.

Papers

- [7] Arkin, R., Towards the Unifition of Navigational Planning and Reactive Control, Working Notes of the AAAI Spring Symposium on Robot Navigation Stanford University, 1989.
- [8] Brooks, R. A., A robust layered control system for a mobile robot, in. IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, pp. 14–23, 1986.
- [9] Pirjanian, P., Behavior Coordination Mechanisms -State of the Art-, http://cres.usc.edu/cgi-bin/print_pub_details.pl?pubid=319 , 1999.
- [10] Steinberg, A. N., Bowman, C. L., White, F. E., Revisions to the JDL Data Fusion Model , ERIM International, Inc. 1101 Wilson Blvd. Arlington, VA 22209, 1999.

Revistas

- [11] Sutherland, I. E., Ebergen, J, Computers without Clocks -Asynchronous chips improve computer performance-, in *Scientific American* July 15, 2002.