Capítulo 3

Sistemas activos

Este capítulo presenta un estado del arte de los sistemas activos. Se describen sus características más importantes y las partes que los conforman. El capítulo está organizado de la siguiente manera. La sección 3.1 introduce a los conceptos de base de los sistemas activos. La sección 3.2 describe el modelo de conocimientos. La sección 3.3 describe el modelo de ejecución. La sección 3.4 presenta la arquitectura de un sistema activo. La sección 3.5 presenta la descripción de dos sistemas activos existentes. La sección 3.6 concluye el capítulo destacando los aspectos más importantes de los sistemas activos.

3.1 Conceptos de base

Un sistema activo es aquel que responde a un estímulo (evento) para producir un efecto en el ambiente o en el interior del sistema donde se observa el evento. La noción de evento se entiende como las señales que provienen de la ejecución de un sistema y de los mensajes intercambiados entre componentes de software. La figura 3.1 muestra la forma en que trabaja un sistema activo.

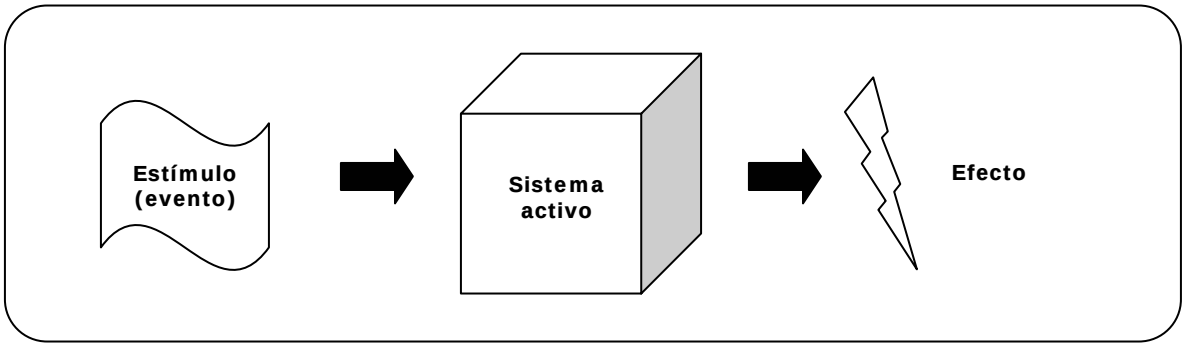

Figura 3.1 Sistema activo

Los sistemas activos se caracterizan por la capacidad de reaccionar automáticamente a ciertos eventos de interés. Estos sistemas comprenden un comportamiento reactivo que está descrito por una relación evento → acción. La expresión de tal comportamiento implica: la definición de los eventos a la acción asociada, la detección de los eventos y la ejecución de las acciones.

Como ejemplo tomemos el sistema de alarma de una casa, el cual, al detectar una situación extraña (apertura de una puerta), tiene la reacción adecuada (informar a la policía).

Los aspectos fundamentales de un sistema activo son:

- La manera en que se detecta un evento, es decir, cómo se detectan situaciones de interés y cómo se descartan aquéllas que son indiferentes. Por lo tanto, es necesario ubicar un módulo que tenga como finalidad observar la aplicación o al proceso del cual nos interesa conocer sus eventos; a este módulo le llamamos monitor.
- Los criterios con los que se evalúa un evento. La evaluación de un evento se realiza por medio de reglas que establecen políticas para la ejecución de acciones.
- La forma de producir un efecto. Una vez que se detecta un evento y se procesa con la regla correspondiente una acción es realizada. La acción comprende la ejecución de una o varias operaciones sobre el proceso o aplicación que produjo el evento o sobre algún otro.

3.2 Modelo de conocimientos

El modelo de conocimientos define la manera de actuar del sistema activo. Establece los esquemas para la detección de un evento y los modelos de evaluación de los mismos.

3.2.1 Modelo de tipos de evento

El modelo de tipos de evento proporciona conceptos y estructuras generales utilizadas para describir tipos de evento. Un tipo de evento es una expresión que describe operaciones ejecutadas por un elemento de software en particular, producidas dentro de un intervalo de tiempo determinado. Este intervalo define el lapso durante el cual los eventos de tipo actual pueden ser detectados. Por lo tanto, los eventos son instancias de un tipo de evento con relevancia para un consumidor [VSCR00]. Un tipo de evento se asocia con aspectos particulares de detección, producción y notificación.

La estructura general que describe un tipo de evento considera el nombre del evento y una variable asociada:

TE_1 = (nombre tipo: string, {(nombre variable: {real, integer, string})})

Un modelo caracteriza tipos de evento como duplas, donde en primer lugar se encuentra el nombre del tipo de evento y en segundo lugar su contenido (conjunto de tuplas que describen el nombre de una variable y su tipo asociado).
Adicional a TE_1 se puede agregar el operador de selección *where* para expresar restricciones en la parte del contenido.

3.2.2 Modelo de administración de eventos

Un modelo de administración de eventos describe cinco fases de administración: definición, detección, producción, notificación e historial.

En la fase de definición los eventos correspondientes a una aplicación dada son descritos usando un tipo de evento. Los procesos de detección y producción monitorean la ejecución

de la aplicación para detectar y reconocer eventos. Estos procesos detectan y producen los eventos descritos previamente en la fase de definición. El proceso de notificación entrega los eventos bajo diferentes modos y protocolos de comunicación, dependiendo de la información necesaria y las características de los consumidores.

Usando un modelo de administración de eventos EM_1 se puede especificar que los eventos sean producidos en un intervalo de tipo [*instante, instante*]. Se puede especificar también que los eventos sean detectados bajo *pull* o *push*[6] de protocolos asíncronos. La notificación puede ser orientada a instancias o a conjuntos, en el primer caso se notifica cada vez que un evento es producido y en el segundo se notifica una vez por cada N eventos producidos. Los eventos son ordenados en un historial y son comparados por sus tiempos de aparición, y persisten ahí hasta que el siguiente evento del mismo tipo llega. Hay que destacar que los valores son escogidos basándose en los requerimientos y características de cada aplicación.

3.2.3 Modelo de reglas activas

Una regla activa establece una asociación entre un evento y una acción. Las reglas activas pueden ser de dos tipos: Evento-Acción (EA) o Evento-Condición-Acción (E[C]A).

Una regla EA está compuesta por un tipo de evento y una acción, mientras que una regla E[C]A es aquella que está compuesta por un tipo de evento, una condición y una acción, donde:

- Tipo evento: especifica uno o más eventos a los cuales se debe reaccionar.
- Condición: especifica una precondición para que la acción sea ejecutada.
- Acción: consiste en una o más operaciones que se deben realizar sobre el sistema.

La estructura de una regla activa de tipo EA es de la forma:

ON *evento*
DO *acción*

La estructura de una regla activa de tipo E[C]A es de la forma:

ON *evento*
IF *condición*
DO *acción*

Para el ejemplo de la sección 3.1, la regla de tipo E[C]A para indicar que el sistema debe llamar a la policía si una puerta se abre entre las 12 y las 6 a.m. es:

ON *aperturaDePuerta = true*
IF *hora > 12 a.m. and hora < 6 a.m.*
DO *llamarPolicia()*

Cuando un evento correspondiente a una regla ocurre se dice que la regla ha sido activada. Si la regla es de tipo EA la acción es ejecutada. En cambio, si la regla es de tipo E[C]A su condición se verifica, y si se satisface la acción es ejecutada. La figura 3.2 muestra la composición de las reglas EA y E[C]A. La figura 3.2 muestra la composición de las reglas EA y E[C]A.

---

[6] Pull indica que la notificación se relaliza de forma inmediata tan pronto como se detecta el evento. Push permite que la notificación se realice al final de un sesión, aunque el evento haya sido detectado previamente.

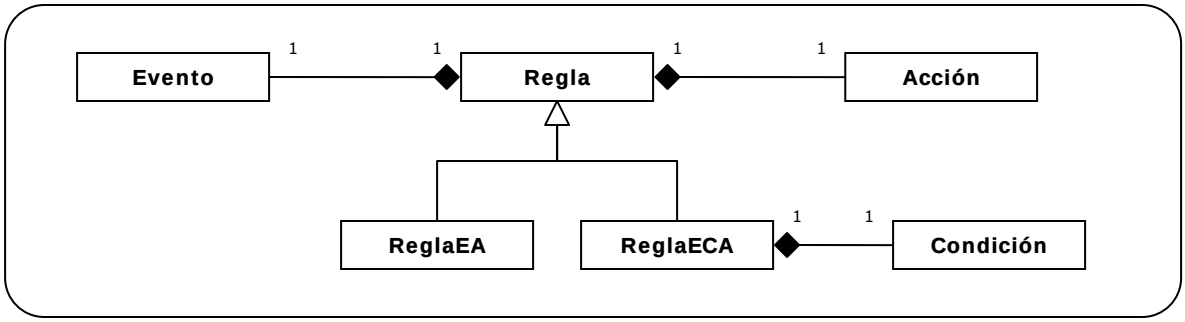

Figura 3.2 Reglas EA y E[C]A

El modelo de reglas activas es paramétrico. Por lo tanto, las aplicaciones pueden adoptar una regla e instanciar sus propiedades de comportamiento de acuerdo a sus características y necesidades [CLD98].

3.2.4 Administración de reglas activas

Un administrador de reglas implementa operaciones de base para el manejo de reglas activas.

Activación y desactivación
Una regla tiene dos estados posibles: activo y no activo. El estado activo corresponde al estado funcional con respecto a los eventos que la disparan. El estado no activo mantiene la regla existente pero de manera pasiva.

Creación de una regla
Una regla es creada a partir de la especificación de sus componentes evento, condición y acción. Su estado inicial a priori es activo, sin embargo, puede ser definido como no activo en su creación.

Actualización de una regla
Esta operación corresponde a la creación de una nueva regla seguida de la destrucción de la regla actual. Los elementos condición y acción de la regla, a diferencia del elemento evento que la define, pueden ser redefinidos.

Destrucción de una regla
La regla es eliminada del administrador. Sólo las reglas desactivadas pueden ser destruidas.

Los administradores de reglas tienen la capacidad de realizar estas operaciones dependiendo de las circunstancias de la aplicación que requiere el servicio.

Los administradores de reglas son capaces de soportar reglas globales y ejecutarlas siguiendo políticas que son adaptadas a las necesidades específicas de la aplicación que lo requiere, proporcionando soporte para la cooperación entre componentes. Dichas políticas de activación de reglas permiten especificar la manera de manejar un evento que ha activado una regla. Para que una regla se verifique, el evento que la activa tiene que ser

considerado dentro de un intervalo de tiempo válido. Un evento puede ser tomado en cuenta tanto para la verificación de una regla como para diferentes verificaciones posteriores al momento de su notificación hasta el fin de su intervalo de tiempo válido. Las reglas pueden ser activadas cada vez que el evento que las dispara es notificado, o sólo una vez para cada activación de un conjunto de eventos.

3.3 Modelo de ejecución

El comportamiento de un sistema activo está representado por el modelo de ejecución de reglas que define las características del tratamiento de las reglas activas durante diferentes fases de su proceso de ejecución.

3.3.1 Proceso de ejecución

El proceso de ejecución de reglas considera dos niveles de abstracción, uno presenta la dinámica de ejecución de una regla activa y el otro proporciona la dinámica de ejecución de un conjunto de éstas.

La ejecución de una regla E[C]A se realiza en tres fases: disparo, evaluación de la condición y ejecución de la acción, mientras que las de una regla del tipo EA son: disparo y ejecución de la acción [GR00].

La figura 3.3 muestra las fases de ejecución de una regla ECA.

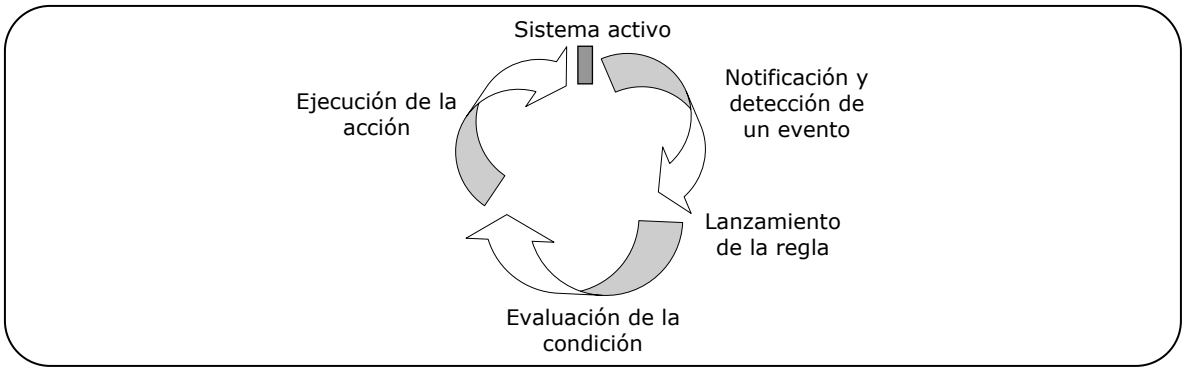

Figura 3.3 Fases de ejecución de una regla ECA

El disparo es el proceso por el cual una regla declarada llega a ser iniciada. Este proceso precede de manera inmediata a la notificación del evento que inicia la regla. Una regla iniciada puede ser evaluada inmediatamente o no. Si la condición de la regla se cumple la acción es ejecutada y se regresa al estado inicial, de otra forma sólo se regresa al estado inicial.

Durante el proceso de ejecución es posible que diferentes reglas sean iniciadas, debido a que un evento inicie varias reglas o porque éstas ya existen en el conjunto de reglas las lanzadas para las cuales es necesario integrar las nuevas reglas iniciadas. La ejecución de

un conjunto de reglas se compone de dos fases: calendarización y selección, las cuales son integradas en el proceso de ejecución de una regla.

La calendarización se rige por criterios definidos por cada sistema activo y es la base para la selección de la siguiente regla a tratar. Está definida por la llegada de eventos, los cuales ocurren en instantes diferentes: ante la ocurrencia de un evento producido por una aplicación, al final de la ejecución de una regla o ante la ocurrencia de un evento producido por la ejecución de una regla.

Una vez que las reglas se han calendarizado es necesario llevarlas a cabo. Una regla es considerada para su ejecución en un instante determinado conocido como punto de ejecución.

3.3.2 Características de ejecución

El modelo de ejecución define lineamientos para la ejecución de reglas activas. Establece cuándo ejecutar una o varias reglas y su relación con el evento que las dispara.

Una regla
Para las tres fases de ejecución de una regla es necesario considerar las características relacionadas con el tratamiento de eventos, la evaluación de la condición y la ejecución de la acción. La evaluación de la condición es iniciada por el evento, y del mismo modo, la ejecución de la acción es iniciada por la evaluación cuando la condición se verifica. Las operaciones ejecutadas sobre los datos por la aplicación son incluidas en transacciones.

Un conjunto de reglas
El modelo de ejecución de un conjunto de reglas determina cómo las reglas iniciadas serán calendarizadas para su ejecución. La ejecución se puede realizar en paralelo o en secuencia. Para la ejecución en secuencia el modelo precisa los criterios a utilizar, por ejemplo, las prioridades en el orden de inicio de las reglas.

3.4 Arquitectura de un sistema activo

Para cubrir los aspectos fundamentales de un sistema activo antes mencionados es necesaria la fusión de dos servicios. El primero de ellos es el servicio de eventos, que involucra la manera en que se detecta un evento. Establece una arquitectura adaptable y extensible, que permite una comunicación basada en eventos. El segundo es el servicio de reglas, que establece los criterios con los cuales se evalúa un evento y la forma en que se debe producir un efecto. La figura 3.4 muestra la arquitectura general de un sistema activo [GR00].

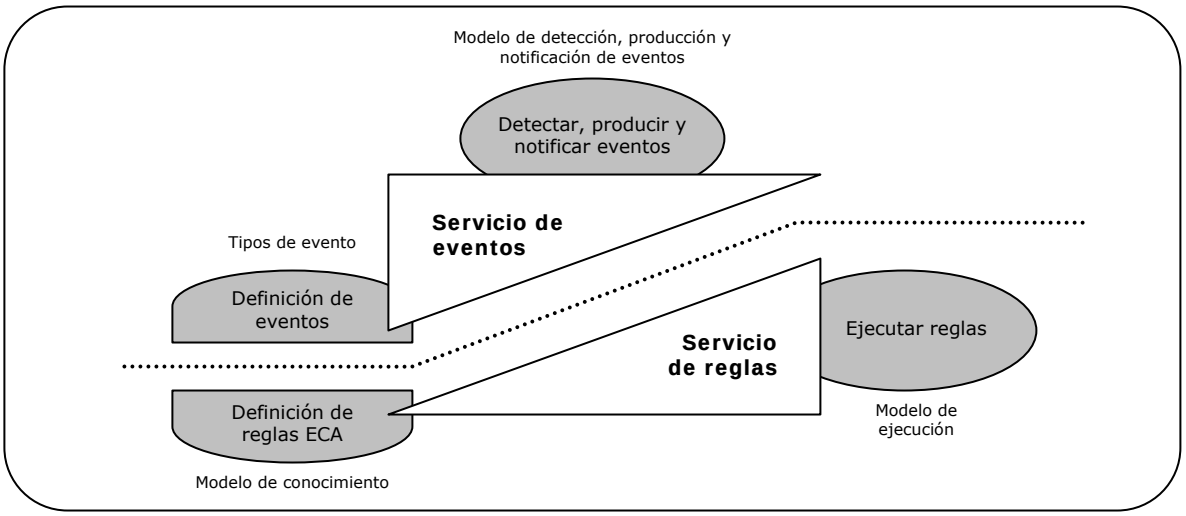

Figura 3.4 Arquitectura general de un sistema activo

3.4.1 Servicio de eventos

Se define un servicio de eventos como un *framework* para la especificación y generación (implementación) de administradores de eventos [VSCR00].

El servicio de eventos soporta un meta-modelo de administración de eventos (ME) y un meta-modelo de tipos de eventos (MTE). El ME está definido en términos de dimensiones asociadas a dominios. Las dimensiones caracterizan diferentes aspectos del procesamiento de las fases de un evento: definición, detección, producción y notificación. Un dominio ofrece los posibles valores para una dimensión. El MTE permite la descripción de tipos de evento simples (nombre y contexto) correspondientes a dominios específicos de la aplicación, así como semánticas de composición de operadores.

Para especificar un administrador de eventos es necesario definir un esquema así como un conjunto de instancias del mismo. Un esquema comprende definiciones de tipo de evento (tE) y un modelo de administración de eventos (mE) para un grupo de clientes. Los tipos de eventos en tE son instancias del MTE. Los mE definidos son instancias del ME que restringe dominios. Por lo tanto, los clientes describen cuáles y cómo los tipos de eventos del esquema serán administrados por el modelo de administración de eventos de tal modo que se definan instancias del esquema. Para un tipo de evento dado, una instancia del esquema se define simplemente por la elección de un valor de su dominio correspondiente por cada dimensión propuesta.

3.4.2 Servicio de reglas

Un servicio de reglas especifica y genera administradores de reglas capaces de cooperar con administradores de eventos (a través de eventos específicos, protocolos de detección y notificación) y para ejecutar reacciones (condiciones y acciones) sobre modelos de ejecución específicos [VSCR00]. Las reacciones son ejecutadas dentro de los marcos de la aplicación. La especificación de un administrador de reglas corresponde principalmente a definir un esquema que engloba un conjunto de reglas activas y un modelo de ejecución.

El servicio de reglas se encuentra implementado por tres meta-modelos: (i) definición de reglas, que proporciona conceptos para la especificación de reglas, (ii) ejecución de reglas, que proporciona dimensiones para caracterizar diferentes aspectos de la ejecución como el consumo de un evento, el modo de ejecución y el modo de transacción, y (iii) cooperación de reglas, que proporciona políticas que describen la manera en que un administrador de reglas coopera con otros componentes tales como un evento, una transacción u otro administrador de reglas.

El servicio de reglas permite caracterizar diferentes aspectos de la ejecución de reglas como consumo de un evento, modo de ejecución, modo de transacciones y múltiples políticas de ejecución [CVSGR00]. La figura 3.5 muestra la arquitectura general del servicio de reglas, donde se incluyen los componentes necesarios para la ejecución de reglas.

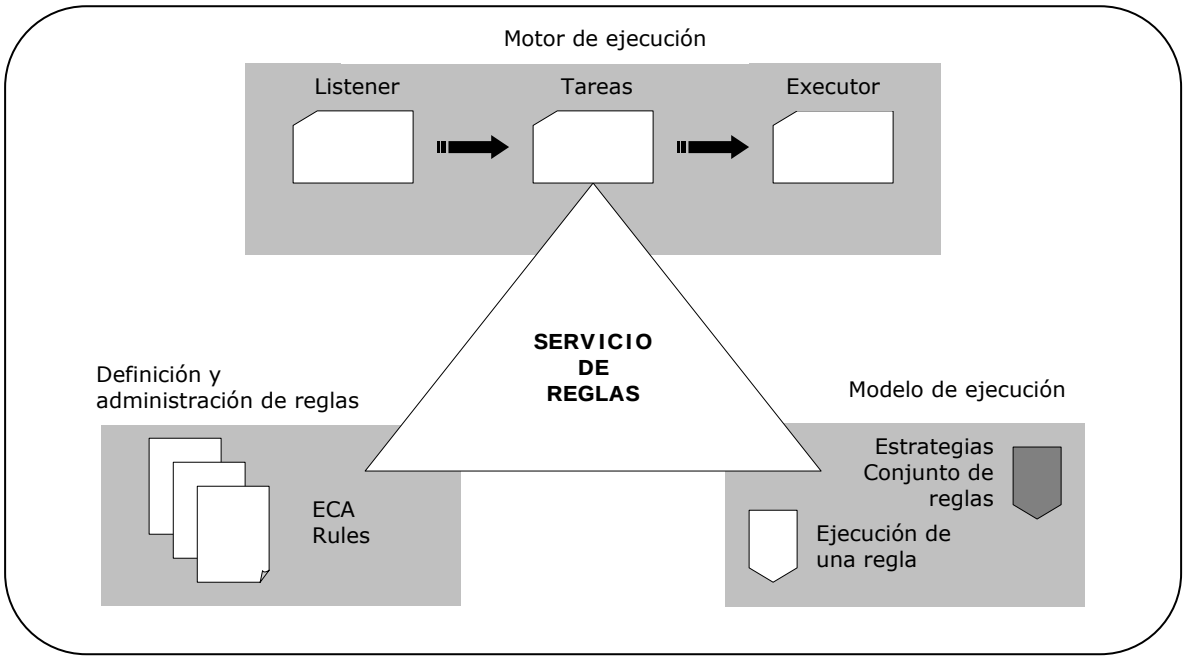

Figura 3.5 Servicio de reglas

El motor de ejecución establece el modelo de ejecución a través de módulos de tareas y ejecutor. Los procesos en el motor de ejecución están constituidos de la siguiente manera: un escucha (*listener*) reconoce los eventos del administrador de eventos, el módulo de tareas ordena las reglas que el evento ha activado y finalmente el ejecutor se encarga de ejecutarlas [GR00].

3.5 Sistemas activos

3.5.1 ADEES

ADEES (*Adaptable and Extensible Event Service*) es una infraestructura basada en eventos que permite desarrollar complejos sistemas distribuidos de información [VSC02].

ADEES puede ser extendido y personalizado en base a las necesidades de una aplicación. ADEES es utilizado para especificar servicios de eventos y generar administradores de eventos encargados de atender tanto a componentes productores como consumidores de eventos, basado en la descripción de eventos y en los modelos de administración de eventos que ellos describen. Más aún, el servicio de eventos y los administradores se reconfiguran y adaptan de manera estática o dinámica al ambiente dependiendo de los requerimientos de la aplicación.

Un administrador detecta la ocurrencia de eventos o las combinaciones de éstos provenientes de los productores y consecuentemente los notifica a los consumidores que han declarado tener interés en ellos. Los administradores de eventos de un servicio comparten la misma manera de describir tipos de eventos y políticas de administración.

3.5.1.1 Arquitectura del servicio de eventos

Un servicio de eventos construido usando ADEES se representa con un servicio administrador de eventos (SAE) y con uno o más administradores de eventos que SAE monitorea. SAE genera administradores de eventos basándose en un esquema (modelo tipo evento y modelo administrador de eventos). Estos administradores pueden ser ocupados por un conjunto de clientes para especificar instancias del esquema mencionado, es decir, tipos de eventos que el cliente produce o consume y sus políticas de administración asociadas. La figura 3.6 muestra la arquitectura del servicio de eventos.

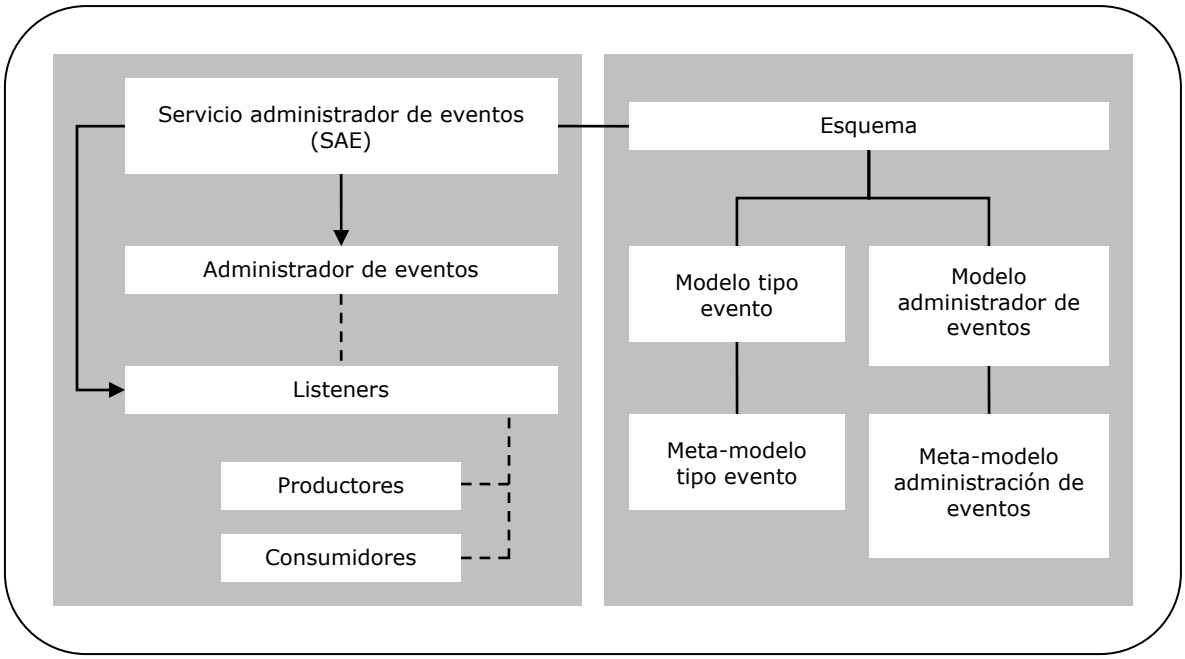

Figura 3.6 Arquitectura del servicio de eventos

SAE no sólo genera administradores de eventos sino que también los supervisa. Proporciona la manera de especificar, guardar y administrar esquemas e instancias, junto con estrategias de desarrollo para los administradores de eventos especificados. Para cada especificación de administradores de eventos, SAE genera componentes para la detección,

producción y notificación de eventos. SAE también define *listeners* (objetos escuchas) que sirven de punto de encuentro entre los administradores y los productores (para la detección de eventos) y consumidores (para la notificación de eventos) suscritos. Basándose en la estrategia de desarrollo especificada, el administrador conecta al cliente con un *listener* nuevo o con uno existente.

Un administrador de eventos está caracterizado por un esquema que le permite aceptar y administrar diferentes instancias definidas por los productores y consumidores suscritos a él. Las instancias definen tipos de eventos en los cuales el administrador está interesado, y políticas de administración asociadas a éstos utilizadas para administrar los tipos de eventos definidos (adaptabilidad estática). Ambos conceptos pueden ser entonces modificados para ser agregados en tiempo de ejecución (adaptabilidad dinámica).

Los administradores de un servicio comparten la misma manera de describir tipos de eventos y modelos de administración. Su esquema se describe con respecto a los mismos meta-modelos de tipo de eventos y de administración de eventos. Un esquema es una instancia de esos meta-modelos y puede ser entonces modificada para extender o reducir las funcionalidades de un servicio de eventos y sus administradores (extensibilidad).

3.5.1.2 Administrador de eventos adaptable

Los eventos producidos y consumidos por un administrador de eventos tienen que ser caracterizados por tipos de eventos y políticas de administración de eventos, respectivamente descritas usando un modelo de tipos de eventos y un modelo de administración de los mismos. Ambos modelos están definidos por un esquema soportado por los administradores de eventos.

Los componentes que desean compartir un esquema deben suscribirse al administrador de eventos correspondiente, ya sea como productores o como consumidores. En la suscripción un cliente especifica una instancia usando los conceptos de los modelos dados en el esquema especificado. Cada cliente suscribe diferentes instancias, una por cada tipo de evento producido o consumido.

3.5.1.2.1 Esquema de administración de eventos

Un esquema soportado por un administrador asocia un modelo de tipo de eventos con un modelo de administración de eventos. Incluye además un modelo de control de comportamiento que especifica diferentes políticas para la activación y desactivación del administrador. Un esquema puede ser definido como E: (TE_1, AE_1) donde TE_1 es un modelo de tipos de evento y AE_1 es un modelo de administración de eventos.

3.5.1.2.2 Instancia

Al momento de suscribirse a un administrador de eventos, tanto productores como consumidores definen instancias de un esquema. Una instancia está conformada por un tipo de evento y por un modelo de administración.

Un tipo de evento está definido como una instancia de un modelo de tipo de evento. Dados un consumidor o productor y un tipo de evento, la correspondiente dimensión debe tener

un valor único. Dicho valor es escogido del dominio asociado a cada dimensión del correspondiente esquema. De esta forma, los productores y consumidores pueden especificar diferentes políticas para ser usadas en la administración de cada tipo de eventos que produzcan y/o consuman.

3.5.1.2.3 Adaptabilidad

La adaptabilidad se denota como la capacidad de un administrador de eventos de adaptar su comportamiento con respecto a un conjunto de clientes. El hecho de proporcionar un esquema y la posibilidad de definir instancias al momento de la suscripción asegura un primer nivel de adaptabilidad para el administrador de eventos. Otro nivel de adaptabilidad se consigue cuando el administrador es capaz de activar, desactivar y actualizar instancias de manera dinámica.

De acuerdo a su tipo las actualizaciones pueden ser inmediatas o diferidas. Por ejemplo, actualizaciones en protocolos de comunicación son ejecutadas inmediatamente, otras dimensiones son modificadas en puntos de comprobación específicos dependiendo del instante de llegada de su solicitud y del estado de ejecución del administrador. Las reglas de actualización que describen las políticas de actualización están definidas en el nivel del servicio y son configuradas al momento de creación del SAE [VSoo].

3.5.1.3 Administradores de eventos extensibles

Conforme las aplicaciones van evolucionando, sus requerimientos con respecto al administrador de eventos pueden incrementarse o sufrir modificaciones. Por lo tanto, debe ser posible especializar modelos implementados por un administrador y autorizar la definición de nuevos tipos de eventos, la especificación de nuevos operadores de composición y de nuevas políticas de administración. De igual forma, actualizaciones en los modelos existentes deben de ser posibles. Consecuentemente los modelos han sido considerados como datos que pueden ser descritos, materializados y administrados por un servicio de eventos. ADEES propone dos meta-modelos que son la piedra angular de su funcionamiento, uno para describir tipos de eventos y otro para describir modelos de administración de eventos.

3.5.1.3.1 Meta-modelos

Meta-modelo tipo evento
Este meta-modelo está conformado por dos meta-clases principales (tipo dato abstracto): *tipoEvento* y operador.

Un tipo de evento está representado por un *TipoEventoNombre* que describe la estructura de los nombres de los eventos, para instanciarlos en un modelo de tipo de eventos los nombres pueden ser *strings* y en otros casos pueden ser expresiones regulares. Un tipo de evento puede ser asociado con información descrita por un conjunto *Delta*. Cada elemento del conjunto es representado por la meta-clase Delta de la forma (*nombreVariable: string, tipoVariable: Tipo*).

Un operador es un tipo de dato que describe el tipo de operaciones que pueden ser asociadas con un tipo de evento. Un tipo de evento caracteriza un operador como una

función descrita por un símbolo, un dominio, un co-dominio y una implementación. El dominio y el co-dominio están definidos como el producto cartesiano de tipos de datos nativos (integer, real, etc.) y tipoEvento. Su implementación puede definirse en un leguaje formal (Ej., Lambda calculus) o en un leguaje de programación específico. En ambos casos es de suponerse que es correcto, y desde el punto de vista de meta-modelos es una caja negra.

Meta-modelo administración eventos
Un modelo de administración de eventos describe cuatro fases de administración: detección, producción, notificación y administración de un histórico. Un meta-modelo de administración de eventos caracteriza dichas fases por dimensiones asociadas a valores. Un valor tiene asociada una implementación que especifica su semántica. Las dimensiones están agrupadas en componentes, donde cada uno representa una fase del administrador de eventos.

El meta-modelo puede ser instanciado para definir diferentes modelos de administración de eventos simplemente especificando los componentes que ellos contienen. Por ejemplo, un modelo de administración puede incluir varias composiciones de eventos, mientras no describen esa funcionalidad. Más aún, dimensiones, valores y sus implementaciones asociadas deben ser también especificadas.

3.5.1.3.2 Extensibilidad

La extensibilidad se denota como la capacidad de un servicio y sus administradores de extender y reducir sus funcionalidades. Es controlada por el SAE, que puede recibir solicitudes explícitas por parte de los clientes o detectar situaciones que provoquen actualizaciones en el esquema.

Un primer nivel de extensibilidad se consigue modificando de manera estática un esquema existente para definir nuevos administradores. Los meta-modelos aseguran una extensión consistente, es decir, aseguran que las actualizaciones en el esquema son hechas con respecto a las dimensiones de los meta-modelos.

Un segundo nivel de extensibilidad se contempla con la posibilidad de modificar de manera dinámica el esquema de un administrador existente. Las actualizaciones pueden ser hechas en modelos de tipo de eventos y en modelos de administración que componen el esquema. Un modelo de tipo de evento puede ser modificado agregándole, borrándole o actualizándole tipos de eventos e implementaciones de operadores.

En el nivel del administrador, las actualizaciones en el esquema implican la activación o desactivación de instancias y la re-configuración de sus componentes internos. Un modelo de administración es actualizado agregándole, modificándole o borrándole valores asociados a sus dimensiones. Tales operaciones implican la modificación interna de los componentes de los administradores y de los *listeners*. De acuerdo con la naturaleza de las actualizaciones, la ejecución de estas operaciones puede ser inmediata o diferida a un punto de comprobación específico. Las políticas para actualizar son especificadas cuando el SAE es configurado.

En el nivel del servicio de eventos la extensibilidad se manifiesta en actualizaciones en los meta-modelos. Por ejemplo, es posible modificar la estructura de tipoEvento (Ej., un cambio de los tipos asociados con los atributos Delta). Igualmente, componentes y dimensiones pueden ser agregadas y eliminadas del meta-modelo de administración de eventos.

3.5.2 ODAS

ODAS (*Open and Distributed Active Services*) es un *framework* que permite especificar y construir sistemas activos distribuidos abiertos, es decir, es utilizado para especificar y generar administradores de eventos y de reglas que soporten la cooperación e interacción entre sistemas distribuidos [CVSGR00].

Los administradores de eventos soportan una comunicación basada en el intercambio de eventos de manera anónima. Los administradores de eventos y los administradores de reglas pueden reconfigurarse y adaptarse a si mismos con respecto a los requerimientos de la aplicación y del medio ambiente. La arquitectura presenta un *framework* que incluye el servicio de reglas y el servicio de eventos. La figura 3.7 muestra la arquitectura de ODAS [CVSGR00].

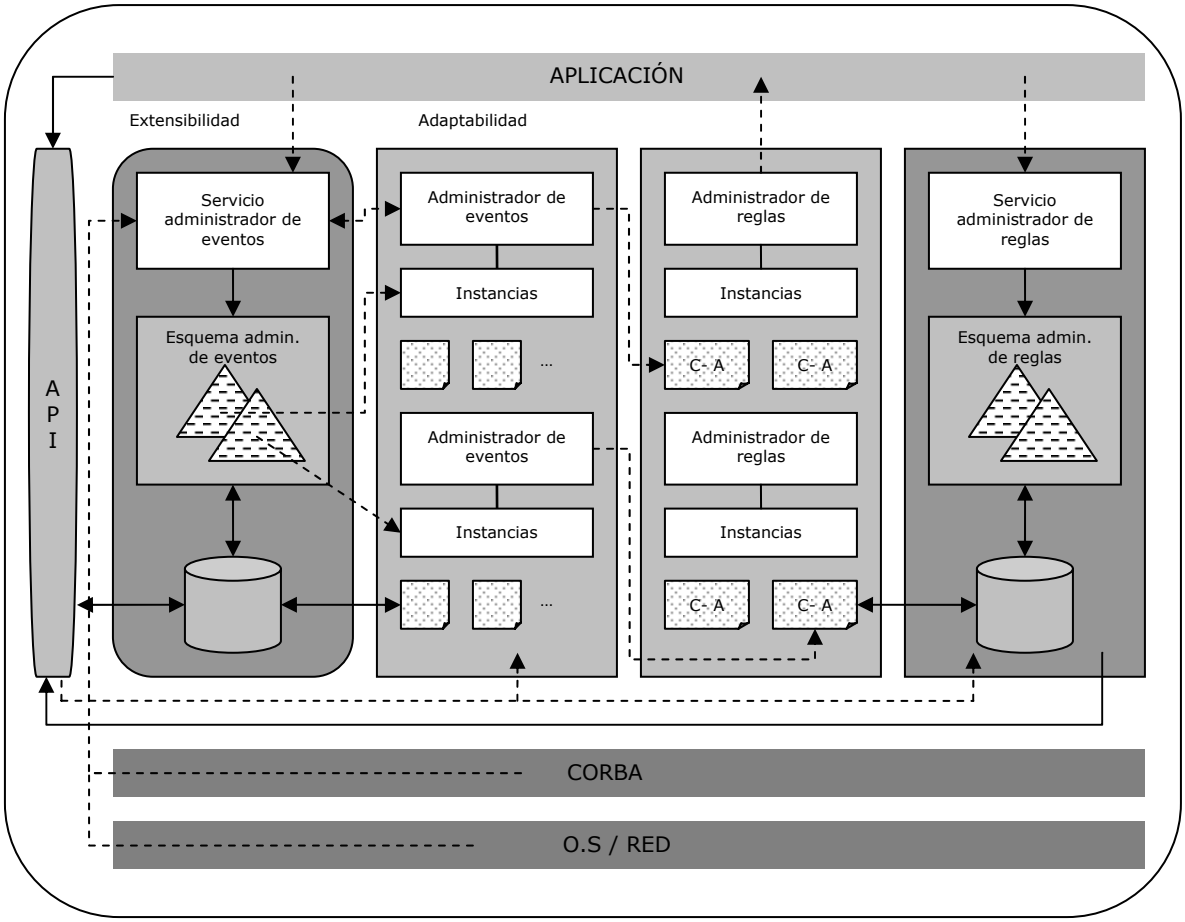

Figura 3.7 Arquitectura de ODAS

Consiste de un API (*Aplication Programming Interfaces*) para especificar esquemas, los cuales describen las necesidades de un grupo de aplicaciones en lo que concierne al manejo de eventos y la ejecución de reglas. Por último se apoya en CORBA para especificar interfaces entre componentes (administrador de eventos, productores y consumidores de eventos, servicios de eventos y de reglas, máquinas de ejecución de reglas). El servicio de eventos de CORBA es utilizado a bajo nivel como infraestructura de comunicación.

ODAS se apoya en ADEES en lo que respecta al servicio de eventos. Adiciona a su arquitectura un servicio de reglas que se basa en el uso de reglas activas. Fusiona ambos servicios para poder generar sistemas activos.

Especificar un administrador de reglas corresponde principalmente a definir un esquema que abarca un conjunto de reglas activas, un modelo de ejecución y un modelo cooperativo. El servicio de reglas soporta tres meta-modelos [CVSGR00]: meta-modelo de definición de reglas, meta-modelo de ejecución de reglas y meta-modelo de cooperación de reglas. La explicación detallada de los meta-modelos se encuentra en la referencia [CVSGR00].

Propiedades

El uso de meta-modelos provee a ODAS de dos propiedades importantes: adaptabilidad y extensibilidad. En la descripción de ADEES fue explicada la forma en que el esquema de administración de eventos colabora en estas dos propiedades. Ahora se describe la forma en que coopera el esquema de administración de reglas.

En cuanto a extensibilidad, el esquema administrador de reglas reúne un conjunto de definiciones de reglas con un modelo de ejecución de las mismas, y el modelo de reglas con uno de cooperación. La definición de las reglas se hace por medio de reglas activas. Un modelo de ejecución de reglas es una instancia del meta-modelo de ejecución de reglas que restringe el dominio con las dimensiones propuestas. Un modelo de cooperación es una instancia del meta-modelo de cooperación. La extensibilidad es soportada por la evolución del esquema.

En cuanto a adaptabilidad, una instancia del esquema administrador de reglas es definida por:

- La asociación de cada dimensión del modelo de ejecución de reglas con uno y sólo un valor de su domino, para una regla del conjunto de definiciones.
- Especificando un conjunto de contactos cooperativos, uno por cada par regla-administrador de eventos, regla-administrador de reglas.

3.6 Conclusiones

Este capítulo presentó el concepto de sistemas activos y la descripción de sus componentes necesaria para entender su funcionamiento. Se introdujo el modelo de conocimientos; el cual establece el tipo de comportamiento de los sistemas activos, correspondiente a la parte esencial encargada de establecer las normas de detección y administración de eventos, así como la definición y administración de reglas activas. Se describió el modelo de ejecución, el cual complementa los lineamientos de operación del sistema activo

agregándole normas para el tratamiento de las reglas activas. Se analizó la arquitectura de los sistemas activos describiendo sus dos componentes principales, el servicio de eventos encargado de la administración de los mismos y el servicio de reglas que comprende los aspectos correspondientes a la ejecución de reglas. Finalmente se analizó el funcionamiento y la arquitectura de dos sistemas activos, ADEES que proporciona una infraestructura basada cien por ciento en eventos para la creación de sistemas distribuidos, y ODAS que toma como base a ADEES en lo referente al servicio de eventos y agrega un servicio de reglas dando como resultado un *framework* que permite la especificación y creación de sistemas activos distribuidos.