Adaptación automática de entornos activos mediante agentes basados en reglas

Xavier Alaman, Abraham Esquivel, Manuel García-Herranz, Pablo A. Haya, Germán Montoro[1]

Departamento de Ingeniería Informática,
C. Fco. Tomás y Valiente, 11. Madrid 28049. Spain
{Xavier.Alaman, Abraham.Esquivel, Manuel.Garciaherranz, Pablo.Haya,
German.Montoro}@uam.es
http://odisea.ii.uam.es

Abstract. La Computación Ubicua presenta sus primeros resultados en la creación de *Entornos Perceptivos.* Por otra parte, el siguiente reto que plantea puede resumirse en cómo transformar estos entornos, capaces de percibir el mundo que los rodea, en *Entornos Interactivos*, en los que sus habitantes puedan comunicarse e enteractuar con él, o en *Entornos Inteligentes* o *Entornos Activos*, capaces de tomar decisiones basadas en esa información de contexto. El problema fundamental que surge a la hora de construir un entorno que se adapte a las necesidades y gustos de sus habitantes radica, precisamente, en la variedad y diversidad de estos. Encontrar un punto común que permita al sistema adaptarse a cada uno de sus diferentes usuarios ha sido centro de estudio de numerosos proyctos. El presente artículo plantea un primer prototipo para un sistema completo que centre la adaptación del entorno en el propio usuario, brindándole los medios necesarios para definir sus deseos y ayudándole en la tarea a través de mecanismos de explicación -que le permitan comprender y modificar el sistema- y de aprendizaje -que refine las preferencias expresadas por el usuario para acercarlas a sus verdaderos deseos. Dicho prototipo se basa en un primer lenguaje basado en reglas que posibilita un primer acercamiento a cada uno de los requisitos del mismo -expresión, explicación y aprendizaje- y facilita la evolución de los mismos. Este enfoque plantea un cambio de *inteligencia* en el entorno desde la *autónoma* más tradicional a una *guiada*, cuyo objetivo es apoyar y potenciar la Inteligencia principal del entorno: el usuario.

1 Introducción

En 1991 Mark Weiser [Weiser, M. 1991] describe un nuevo mundo en el que la computación se desvincula del ordenador tradicional para penetrar en los objetos cotidianos, diluyéndose así en el mundo natural del hombre y desapareciendo, a sus ojos, como tecnología desligada del entorno. Desde entonces numerosos avances han ido dando forma a lo que se ha dado en llamar *Computación Ubicua*, *Entornos Activos* o *Entornos Inteligentes* y numerosos retos han visto la luz en el proceso de integrar la computación en la realidad del hombre en lugar de al hombre en la de la computación.

El primer reto que plantea la construcción de este espacio computacional es la creación de un marco común de entendimiento mediante el cual los distintos elementos del entorno puedan definirse y comunicarse. Persiguiendo este objetivo, desde la Universidad Autónoma de Madrid y bajo el proyecto Interact [Haya, P.A. et al. 2004] se ha desarrollado una arquitectura de pizarra capaz de dotar al entorno de esta visión global, aglutinando y homogeneizando la información aportada por los distintos dispositivos del entorno. Esta Capa de Contexto ha posibilitado la aparición de numerosas aplicaciones sensibles al contexto (i.e. interfaces orales con desambiguación contextual, marcos fotográficos que se adaptan a los habitantes…) capaces de sacar provecho de esta visión ampliada del entorno en beneficio del usuario.

Una vez definido este marco unificador que permite a los distintos elementos del entorno conocerse y comunicarse, uno de los retos más interesantes que se plantea es cómo transformar estos *Entornos Perceptivos*, capaces de percibir el mundo que los rodea, en *Entornos Interactivos*, en los que sus habitantes pueden comunicarse e interactuar con él, o en *Entornos Inteligentes* o *Entornos Activos*, capaces de tomar decisiones basadas en esa información de contexto. En otras palabras, se plantea la pregunta: ¿cómo poblar este espacio con aplicaciones sensibles al contexto que actúen en beneficio del usuario? De hecho, el beneficio del usuario es una cuestión de contexto, ligado a quién es este usuario y en qué tipo de entorno se encuentra. De esta forma resulta fácil apreciar que una persona mayor viviendo en casa o una joven en el trabajo comparten poco más allá del deseo de confort y felicidad, ambos términos ambiguos con distinto significado para cada una de ellas.

---

1 Autores por orden alfabético

Así, teniendo la información de contexto y la habilidad para modificarlo, es necesario conseguir la capacidad de definir los deseos y preferencias de sus habitantes; pero ¿tiene el propio habitante esta capacidad? Podemos decir que todo el mundo sabe qué quiere o qué no quiere pero, como en el cuento del genio de la lámpara, saber y definir no siempre son la misma cosa. Se puede apreciar ya, llegados a este punto, que la tarea de adaptar el entorno a las preferencias del usuario no será trivial.

1.1 Diferentes usuarios – diferentes preferencias: el modelo de Agente

En un grado mucho mayor que en el mundo computacional tradicional, este nuevo mundo rico en componentes, información y capacidad de interacción permite la creación de numerosas aplicaciones capaces de sacar provecho del conocimiento del contexto para proporcionar diferentes servicios. Estas aplicaciones pueden ser tan variadas y complejas como la localización de objetos perdidos o el seguimiento de personas mayores [Kidd, C.K. et al. 1999]. Todas estas aplicaciones pertenecen al grupo de las *context-aware applcations* y son, por norma general, desarrolladas para solucionar un problema en particular.

Dado que no todos los usuarios tienen los mismos problemas ni desean afrontarlos de igual modo estas aplicaciones se centran en los problemas más frecuentes y, en algunos casos, permiten personalizar sus soluciones mediante una serie de parámetros que el usuario puede definir.

Pero existe un amplio conjunto de cuestiones más simples y personales que también necesitan ser abordadas en un entorno interactivo. Dichas cuestiones son lo suficientemente variadas, sencillas y dependientes de las preferencias del usuario como para que la implementación de soluciones ad-hoc, demasiado específicas y numerosas, resulte rentable pero, por otro lado, son precisamente estos problemas los que conforman el día a día de la persona. Se hace necesario, por tanto, dotar a los *entornos activos* de algún mecanismo mediante el cual los diferentes usuarios puedan comunicar sus preferencias sobre cómo abordar este conjunto de variadas cuestiones cotidianas, diseñando ellos mismos sus propias aplicaciones.

Centrándose en este problema de adaptación, que podría entenderse como el primer paso entre los *Entornos Perceptivos* y los *Inteligentes*, el presente artículo se centra en la búsqueda de una solución mediante la cual dotar al usuario de la capacidad de modelar el entorno a su gusto. En contraste con el concepto más extendido de inteligencia, no se pretende si no que ésta resida en el usuario, no siendo el entorno sino una herramienta que le permita superar sus barreras físicas y temporales, extendiendo su capacidad de acción y control. Así, la "inteligencia" del entorno no es más que un complemento de la del usuario, verdadera Inteligencia y motor principal, como se verá a lo largo de este trabajo.

Este nuevo concepto de *Entorno Inteligente* trae consigo las ventajas e inconvenientes de la interacción. Dado que, hasta el momento, el único elemento capaz de interactuar con el entorno era el hombre, el hecho de que nuevos agentes lo pueblen y modifiquen puede hacer sentir a éste que el dominio de su propio espacio está amenazado. Es por esto que, como ya anunciara P. Maes "existen dos problemas que deben ser resueltos a la hora de construir agentes software: competencia y confianza" [Maes, P. 1994]. Así como el primero es un factor corrientemente tenido en cuenta en el desarrollo de software, el segundo tiene un carácter más social, menos estudiado en el desarrollo de aplicaciones, surgido del sentimiento de inseguridad fruto de la "invasión" de su espacio y de especial relevancia en el problema que nos ocupa.

2 Prototipo

Una vez determinados los principios de diseño del sistema y disponiendo tanto de un entorno real de pruebas como de una capa middleware que soluciona los problemas de comunicación y acceso a la información, ambos desarrollados durante el proyecto Interact [Haya, P.A. 2006], se ha procedido a la implementación de un primer prototipo que aborda, en fase inicial, cada uno de los requisitos del sistema: *facilidad de expresión*, *capacidad de explicación* y *aprendizaje automático*, así como sus máximas de *no-intrusión* y *modularidad*.

A continuación, y a lo largo de los distintos puntos de esta sección, se describe el estado de cada uno de los aspectos anteriormente citados, ilustrándolos con código y ejemplos que ayuden a comprender la naturaleza real del sistema.

En este sentido, y antes de proceder, es importante señalar que este primer prototipo, desde sus comienzos, ha ido dando cobertura a los servicios más básicos, y a otros no tanto, necesarios en el entorno de pruebas sobre el que se implementó, así como a otros despachos que se han ido añadiendo a la

experiencia domótica que tanto éste como otros trabajos van dando forma. El entorno de pruebas está compuesto de una zona de estar de tipo salón que alberga reuniones de diversa índole, así como momentos de estudio o lectura más relajados y personales, además de un par de mesas de trabajo, ocupadas esporádicamente por distinto personal. Por otra parte, una segunda zona alberga un espacio de oficina con dos puestos de trabajo permanentes. Dicho entorno está poblado de diversos elementos domóticos, controlables o sensibles, como luces, interruptores, enchufes, cafetera, televisión, cerradura y sensor de puerta, lector de tarjetas, radio, altavoces o los propios ordenadores de trabajo que lo convierten en un entorno con altas posibilidades de interacción, ideal para los propósitos de este trabajo de investigación.

2.1 Lenguaje

La unidad mínima completa del lenguaje es la regla, pieza fundamental del conocimiento que encapsula una preferencia del usuario o una inferencia contextual, esto es, una reacción deseada en respuesta a cierta situación emergente o bien información contextual de alto nivel inferida a partir de esa misma situación. Cada regla queda dividida en tres partes **detonantes**, **conclusiones** y **acción** que, a medida que el sistema ha ido creciendo y demandando mayor potencia descriptiva, han ido desarrollando y ampliando su vocabulario.

Dos factores han sido decisivos a la hora de optar por definir un nuevo lenguaje basado en reglas: sencillez y adecuación. La primera por cuanto facilita la tarea de trasladar el sistema al terreno de interfaces más complejas como por ejemplo, la interacción mediante lenguaje natural, permitiendo una traducción mucho más sencilla de un dominio al otro. La segunda en tanto que la idiosincrasia del problema que se trata (expresar deseos humanos sobre entornos físicos) requiere de características y conceptos muy particulares, como los detonantes, que si bien definibles en lenguajes más completos como JESS o CLIPS lo son de forma mucho más compleja. Por otra parte estos lenguajes, a día de hoy y con la complejidad que se le exige al incipiente prototipo, presentan características más potentes de lo que se requiere en detrimento de una simplicidad que, por el momento, sí es necesaria.

Es preciso destacar que el mundo descrito en Interact [Haya, P.A. et al. 2004] se basa en los conceptos de *entidad*, pieza fundamental de su estructura, *propiedad*, que determina una característica propia de la naturaleza de la *entidad* a la que pertenece -pudiendo variar su valor pero no su existencia- y, finalmente, la *relación*, que viene a representar un vínculo entre dos *entidades*. Estos tres elementos son los que describen el contexto del entorno inteligente, representando entidades reales o virtuales con sus características y vínculos y serán por tanto el marco común de entendimiento entre el usuario y el sistema, ya que representan la parte del mundo del primero que el segundo es capaz de abarcar.

De esta manera una regla queda definida por la gramática de la Figura 1:

```
<regla>:= <lista detonantes> :: <lista condiciones> ->
<acción>

<lista detonantes> := <detonante> | <detonante> ; <lista
detonantes>

<detonante> := <propiedad> | <relación>

<lista condiciones> := <condición> | <condición> ; <lista
condiciones>

<condición> := <elemento> <operador> <elemento>

<elemento> := <entidad> | <propiedad> | <relación> | <valor>

<operador> := = | ¡= | > | <

<acción> := <elemento> <operación> <elemento>

<operación> := := | =¬ | += | -= | => | <=
```

Figura 1. Gramática de una regla del lenguaje del sistema

Como cabe imaginar, no todas las condiciones o acciones sintácticamente correctas son semánticamente válidas. De esta manera el sistema sólo permite las agrupaciones especificadas en las tablas 1 y 2 en donde se puede observar tanto el valor semántico como las combinaciones válidas de operadores y operaciones, respectivamente.

tipo	nombre	significado	LHS	RHS

operador	=	Es igual a	propiedad	valor
				propiedad
			relación	entidad
				relación
	¡=	Es distinto a	propiedad	valor
				propiedad
			relación	entidad
				relación
	>	Es mayor que	propiedad	valor
				propiedad
	<	Es menor que	propiedad	valor
				propiedad

Tabla 1: Significados y configuraciones de los diferentes operadores, donde RHS se refiere al elemento a la derecha del operador y LHS al de la izquierda: <LHS> <operador> <RHS> de una condición

tipo	nombre	significado	LHS	RHS
operación	:=	Asignar valor	propiedad	valor
				propiedad
	=¬	Asignar valor contrario	propiedad	valor
				propiedad
	+=	Incrementar en	propiedad	valor
				propiedad
	-=	Decrementar en	propiedad	valor
				propiedad
	=>	Crear relación	relación	entidad
	<=	Borrar relación	relación	entidad

Tabla 2: Significados y configuraciones de las diferentes operaciones, donde RHS se refiere al elemento a la derecha de la operación y LHS al de la izquierda: <LHS> <operación> <RHS> de una acción

2.2 Agente

Tan importante como estas unidades mínimas o reglas que conforman los ladrillos del sistema es el agente que podría equipararse, siguiendo la analogía con la construcción, con el edificio independiente. De esta manera un agente esta constituido por un conjunto de reglas. En base a ellas, y a los cambios producidos en el contexto, reacciona produciendo una salida en forma órdenes o cambios a realizar sobre ese mismo contexto. La forma en que reacciona a esos cambios, las consecuencias internas que conllevan así como el proceso de inferencia que detonan puede representarse en forma de algoritmo.

Tal y como muestra la Figura 2 el algoritmo de ejecución del agente consta, fundamentalmente, de tres pasos: *educación*, *actualización* y *detonación* que se encadenan en respuesta a un cambio en el entorno.

La *educación* o aprendizaje, que se explicará en más detalle en el apartado 2.5, consiste en utilizar la información correspondiente al cambio producido en el contexto para potenciar o castigar las reglas que se ejecutaron de manera correcta o errónea respectivamente.

La *actualización* consiste en calcular el nuevo valor –verdadero o falso- de todas las condiciones que hagan referencia al elemento contextual objeto del cambio. Esta política permite tener la información que el agente necesita para la toma de decisiones siempre actualizada. Una segunda alternativa pudiera consistir en consultar el valor de cada condición exclusivamente cuando la regla a la que pertenece fuera detonada. Esta alternativa presenta una carga menor en las comunicaciones del sistema ya que sólo requiere la información cuando es estrictamente necesaria. Se ha elegido en cambio la primera, con su aparente sobrecarga de las comunicaciones del sistema por dos razones: primeramente, y menos importante, a medida que el número de reglas crece en un agente, el número de elementos que pertenece al conjunto de las condiciones sin pertenecer al de los detonantes se acerca a cero. Esto quiere decir que a medida que el número de reglas de un agente se incrementa, el flujo de comunicación supuestamente innecesario se vuelve imprescindible y, sacar provecho del mismo parece una buena estrategia. En segundo lugar, y mucho más importante, uno de los puntos más críticos en la adaptación de entornos reales se encuentra en el tiempo de reacción del sistema de automatización. Así, por ejemplo, un retardo de 500 ms al encender una luz provoca en el usuario la sensación de que el sistema está fallando. De esta manera, al actualizar el valor de las condiciones a medida que se producen los cambios en el contexto puede sobrecargar levemente las comunicaciones, pero distribuye el esfuerzo computacional a lo largo del tiempo, agilizando el comportamiento del sistema cuando es más necesario, esto es, a la hora de tomar una decisión.

Por último, la *detonación* consiste en, dadas las reglas que como detonante incluyen el elemento objeto de cambio en el contexto, comprobar que todas sus condiciones evalúen a verdadero y, en su caso, enviar la acción asociada a la pizarra.

```
Para cada cambio en el contexto
    Se educa al agente
    Se actualiza el valor de las condiciones de las reglas
    Se obtienen las reglas detonadas
      Si sus condiciones evaluan a verdadero
        Se envían sus acciones a la pizarra
```

Figura 2. Algoritmo cualitativo de ejecución de un agente

2.3 Expresión

Sacando provecho de la sencillez del lenguaje definido en el apartado anterior, así como de la naturalidad inherente a los lenguajes de reglas para describir comportamientos, se ha dado un primer paso en la consecución del requisito de *facilidad de expresión* usando la expresión escrita.

Para ello se ha implementado un sistema de carga de ficheros. Estos ficheros pueden contener tanto reglas, escritas en el formato anteriormente citado, como comentarios que sirvan de aclaración al usuario. Estos últimos son definidos con el caracter de inicio #. Por otra parte el sistema también permite guardar las reglas de un agente en un fichero de manera que la importación y exportación de paquetes de comportamientos siga un esquema común al de muchas aplicaciones informáticas y facilite la distribución y almacenamiento de paquetes de comportamientos.

Este sistema permite añadir comportamientos al entorno (ver Ejemplo 1), configurar el comportamiento de dispositivos (ver Ejemplo 2) o describir contexto de alto nivel a partir de bajo nivel (ver Ejemplo 3) con forma altamente natural y en pocos minutos.

```
#Luces y televisión
    #Regla 1: cuando se enciende la TV se apaga la luz del techo
    tv:eib6:status :: tv:eib6:status = 1 -> light:lamp_1:status := 0
    # Regla 2: cuando se apaga la TV se enciende la luz del techo
    tv:eib6:status :: tv:eib6:status = 0 -> light:lamp_1:status := 1
   # Regla 3: cuando se enciende la TV, si la lámpara regulable esta
```

```
alta se baja a media

    tv:eib6:status :: tv:eib6:status = 1 ;
dimmablelight:lampv1:value > 20 -> dimmablelight:lampv1:value := 20

   # Regla 4: cuando se apaga la TV, si la lámpara regulable esta
media se sube un poco

    tv:eib6:status :: tv:eib6:status = 0 ;
dimmablelight:lampv1:value = 20 -> dimmablelight:lampv1:value := 35
```

Ejemplo 1. Gestión de las luces en función del encendido y apagado de la televisión. Sólo hay un detonante que es la televisión. Las condiciones sirven para: determinar si la televisión se enciende o se apaga y, en segundo lugar, establecer otras cláusulas necesarias como en las reglas tercera y cuarta.

```
#Interruptores

    switch:switch1_up:value :: -> light:lamp_2:status := 1

    switch:switch1_down:value :: -> light:lamp_2:status := 0

    switch:switch2_up:value :: -> light:lamp_1:status := 1

    switch:switch2_down:value :: -> light:lamp_1:status := 0
```

Ejemplo 2. Configuración de los interruptores swich_1 y switch_2 para que apaguen y enciendan las luces lamp_1 y lamp_2 respectivamente con sus sub-interruptores superior e inferior swittchX_up y switchX_down. Se puede observar que el detonante es la parte superior o inferior del interruptor y que la sección de condiciones está vacía

```
#Inferencia de la localización de Germán en función del estado de su
#ordenador (Indira)

     #Si Indira está ocupado, Germán está en su despacho

     pc:indira:busy :: pc:indira:busy = 1 -> person:german:locatedAt
=> room:lab_b349

     #Si Indira no está ocupado, Germán no está en su despacho

     pc:indira:busy :: pc:indira:busy = 0 -> person:german:locatedAt
<= room:lab_b349
```

Ejemplo 3. Inferencia de la localización de Germán a partir del estado de su ordenador. La acción consiste en crear o destruir, respectivamente, la relación LocatedAt entre Germán y su despacho, el lab_b349.

Complementando a la facilidad de expresión se encuentra la potencia del lenguaje. Mediante el mismo mecanismo de reglas en ficheros se han añadido comportamientos avanzados, equivalentes a aplicaciones sensibles al contexto, en pocas reglas y tiempo de desarrollo. Así, por ejemplo, se ha configurado un interruptor para que encienda o apague todas las luces progresivamente a medida que el usuario siga apretando su parte superior o inferior respectivamente. Es decir, el interruptor deja de estar asociado a una luz en particular para pasar a ser una interfaz que el usuario puede usar para *requerir más o menos luz* en la habitación (ver Ejemplo 4)

```
#Configuración avanzada del interruptor

   #Si se la luz del techo está ya encendida se enciende la de
lectura de la derecha

     switch:switch1_up:value :: light:lamp_1:status = 1 ;
dimmablelight:lampv1:status = 0 -> dimmablelight:lampv1:status := 1

   #Si tanto el techo como la de lectura de la derecha están
encendidas, se enciende la de lectura de la izquierda

    switch:switch1_up:value :: light:lamp_1:status = 1 ;
dimmablelight:lampv1:status = 1 ; dimmablelight:lampv2:status =0 ->
dimmablelight:lampv2:status := 1

   #Si la luz del techo ya está apagada, se apaga la luz de lectura
```

```
de la izquierda

    switch:switch1_down:value :: light:lamp_1:status = 0 ;
dimmablelight:lampv1:status = 1 -> dimmablelight:lampv1:status = 0

#Si tanto el techo como la luz de lectura de la derecha ya están
apagadas, se apaga la luz de lectura de la izquierda

    switch:switch1_down:value :: light:lamp_1:status = 0 ;
dimmablelight:lampv1:status = 0 ; dimmablelight:lampv2:status = 1 ->
dimmablelight:lampv2:status := 0
```

Ejemplo 4. Configuración de un interruptor para que encienda y apague progresivamente las luces de la habitación a medida que es pulsado por el usuario. Adicionalmente esta configuración permite obtener cualquier disposición de luces en la habitación con un mecanismo similar al de las *Torres de Hanoi*

Por último, el sistema implementa una interfaz de tipo consola que permite al usuario visualizar las reglas cargadas para activarlas o desactivarlas mediante un comando de menú. Esto permite reconfigurar más rápidamente un agente que ya se encuentre en uso, anulando comportamientos no deseados o recuperando otros que vuelven a serlo.

2.4 Explicación

A través de la misma consola, el primer paso para conseguir la *capacidad de explicación* enunciada por el segundo requisito, consiste en un sistema de trazas, que pueden ser activadas y desactivadas por el usuario, y que muestran el proceso interno del agente –que comprende los tres pasos enunciados en el apartado 2.2 más la propia "ejecución" del agente en respuesta al cambio en el contexto-.

Dado que el comportamiento de un agente depende enteramente de estos cuatro factores, a saber: en respuesta a qué evento actúa, cómo ha sido educado, cuál es el valor de las condiciones de las reglas y qué regla ha sido detonada, el sistema de explicación debe permitir al usuario vislumbrar el desarrollo y estado de dichos factores. Así, el sistema de trazas, del que se hacía mención al comienzo de este apartado, muestra al usuario la información mencionada (ver Figura 3) en caso de que éste haya decidido activar las trazas en el menú de consola.

```
- Changed property: tv:eib6:status to value: 1

    • Cambio esperado, se educa la regla 32

        * Confianza de la acción:

          tv:eib6:status := 1 (9)

          incrementado, nuevo valor de confianza: 10

[ ] Rule 32: coffeemaker:eib3:status :: coffeemaker:eib3:status = 1
-> tv:eib6:status := 1 (10)  reeducada

        *****Elemento: tv:eib6:status******

    • Actualiza condiciones en las reglas:

        - [ ] Rule 27: tv:eib6:status :: tv:eib6:status = 1 ->
light:lamp_1:status := 0 (10)

                - tv:eib6:status = 1 (true)

        - [ ] Rule 28: tv:eib6:status :: tv:eib6:status = 0 ->
light:lamp_1:status := 1 (10)

                - tv:eib6:status = 0 (false)

    • Detona las reglas:

        - [ ] Rule 27: tv:eib6:status :: tv:eib6:status = 1 ->
light:lamp_1:status := 0 (10)

        - [ ] Rule 28: tv:eib6:status :: tv:eib6:status = 0 ->
light:lamp_1:status := 1 (10)

    • Reglas aplicadas:

        - [ ] Rule 27: tv:eib6:status :: tv:eib6:status = 1 ->
light:lamp_1:status := 0 (10)

Propiedad light:lamp_1:status puesto a 0
```

Figura 3. Información mostrada por el agente cuando todas las trazas están activadas. Primeramente se muestra la educación de las reglas que modificaron la variable que ha cambiado y aun no han sido educadas. En segundo lugar se muestra la actualización de condiciones que contengan la variable cambiada, en todas las reglas. En tercer lugar se muestran las reglas detonadas. Finalmente se muestran las reglas detonadas que cumplen todas las condiciones y por lo tanto son aplicadas. En último lugar aparecen los cambios enviados a la pizarra.

Esta primera aproximación a un sistema de explicación reúne toda la información necesaria, sirviendo de base para la futura implementación de mecanismos más sofisticados que "digieran" dicha información para el usuario, proporcionándola en un formato más natural, o lidien de manera más dinámica con las preferencias de intrusión del usuario (ver apartado 2.6).

2.5 Aprendizaje

En relación al tercer requisito, *aprendizaje automático*, se ha implementado un algoritmo de aprendizaje por refuerzo que usa como tal las acciones llevadas a cabo en el entorno, tanto por el usuario como por otros agentes o interfaces. De esta manera cada regla tiene un peso asociado, equivalente al *factor de confianza*. Cada vez que una regla es detonada, y su acción ejecutada, queda marcada para ser reeducada. Si antes de cierto tiempo o *periodo de penalización* (actualmente fijo en un minuto) la acción ejecutada por la regla no es contradicha, esto es, el valor de la entidad, propiedad o relación modificada no cambia, el peso de la regla se incrementa con un *factor de recompensa* (actualmente se incrementa en una cantidad fija: 1); en caso contrario el peso de la regla se decrementa con un *factor de penalización* (actualmente, al igual que con el de recompensa, se trata de una cantidad fija que, también, es 1). Los valores actuales, altamente primitivos son suficientes para ponderar y probar el sistema, variarlos y ajustarlos es una tarea que requerirá estudio en un futuro. De esta manera se entiende que cuando una regla ejecuta una acción no deseada, el usuario la corregirá en un tiempo determinado, que tiende a ser breve, tras el cual se entiende que cualquier cambio en el contexto, y más concretamente en el objeto de la acción que ejecutó la regla, es independiente de su grado de corrección.

Así, los pesos o *factores de confianza* de las reglas se ven incrementados y/o decrementados a lo largo de su existencia. Este movimiento es supervisado de manera que cuando se cae por debajo de un cierto *umbral de desactivación* o se supera un *umbral de activación* (actualmente ambos iguales a 0) la regla en

cuestión entra, respectivamente, en una *rutina de desactivación y de activación* que serán descritas en más detalle en el siguiente apartado.

Cabe destacar que, así como una regla activa se refuerza negativamente en función de las acciones que toma y no debió tomar, una regla inactiva se detona de igual forma que su homóloga activa con la particularidad de que no envía su acción a la pizarra y, de esta forma, se refuerza positivamente en función de las acciones que debió tomar y no tomó.

2.6 Máxima de no intrusión

La primera máxima para el paradigma de agente planteado hace referencia a las interrupciones que recibe el usuario por parte del sistema y a cómo éstas deben limitarse exclusivamente a lo que aquél esté dispuesto a tolerar. Queda claro que a la hora de notificar, requerir o, en definitiva, interrumpir al usuario, no hay estrategia buena más allá de la específica para cada individuo e incluso ésta puede garantizar poco, ya que un exceso de comunicación molesta y su defecto suele conllevar ineficiencias. [Hamill, L. and Harper, R. 2006] resalta este hecho de las relaciones amo-siervo de una casa inglesa del periodo Victoriano y, de un modo u otro, plantea el dilema eficiencia-comodidad que surge de la comunicación del entorno (en el que se incluyen los agentes y siervos) con el usuario o amo.

Este problema aflora por vez primera cuando una regla entra en la *rutina de activación (o desactivación)* En este momento, el agente ha localizado una regla que, o bien estando desactivada cree que debería activarse o viceversa. Llegado a este punto el agente tiene dos opciones: consultar al usuario que, por tratarse de sus deseos, tomará la mejor decisión, incrementando la eficiencia o activar o desactivar la regla –según corresponda- sin consultar al usuario, incrementando la comodidad o, mejor dicho, no molestando. Dado que este problema no tiene solución correcta sino que depende de cada usuario, el agente debería poder lidiar con los perfiles de cada uno de ellos para decidir cuál de las dos alternativas tomar. Como una primera aproximación a las preferencias del usuario, la consola del agente permite la activación y desactivación de interrupciones, de manera que en el primer caso la rutina consiste en informar al usuario del fenómeno observado y esperar que aclare la situación mientras que en el segundo simplemente actúa, activando o desactivando la regla, sin dirigirse al usuario. Esto personaliza el agente explícitamente y de forma global, siendo deseable, en un futuro, alternativas dinámicas –que cambien su política de interrupciones en función del contexto, sin orden explícita del usuario- y personales, adaptándose a cada usuario. Se puede observar que ambos puntos definen una preferencia en función del contexto –cómo están las cosas o quién está ahí- por lo que se intuye que el mismo sistema de agentes pueda servir para adaptar la política de interrupciones de otros agentes, o de ellos mismos.

2.7 Arquitectura

En relación con la arquitectura modular es necesario destacar algunos puntos. En primer lugar, como ya se dijo, las preferencias, comportamientos e inferencias que codifican las reglas pueden, y deben, ser distribuidas en diferentes agentes, que operan de forma independiente. Por otra parte estos agentes no dejan de ser entidades que, aunque virtuales, son capaces de modificar el entorno, manifestando una presencia tan real que se vuelve necesario disponer de una representación mediante la cual no sólo se puedan observar las acciones sino también los agentes en sí mismos, de igual manera que se puede ver si en un cierto momento el jardinero está o no en el jardín y no sólo los efectos que produce.

En consecuencia, todo agente dispone de una representación en la pizarra que, actualmente, consta de un identificador o *nombre* y un *estado*. De está forma se puede saber no sólo qué agentes están presentes en el entorno si no también en que estado se encuentran. Adicionalmente esta propiedad *estado* permite que el agente sea activado y desactivado desde el exterior, proporcionando las herramientas necesarias para que otros agentes o *meta-agentes* se encarguen de la activación y desactivación del mismo en función del contexto.

2.8 Un banco de pruebas

Tal y como se ha enunciado anteriormente, el sistema no sólo ha sido puesto a prueba en un entorno real sino que permanece en ejecución de manera casi constante, dando soporte a un buen número de servicios de primera necesidad como son el control de acceso o de luces, así como a otros más secundarios.

Dada la estructura modular de los agentes no se puede hablar de la configuración del sistema ya que ésta cambia con el tiempo, a medida que las preferencias evolucionan o más gente decide hacer uso de los agentes, añadiendo, eliminando o modificando los ya existentes, si no de configuraciones puntuales o

habituales. Actualmente, la configuración vigente, consta de seis agentes que sirven bien de ejemplo para ilustrar las diferentes potencialidades del sistema. Estos seis agentes se encargan, respectivamente de: **el control de luces**, **el control de acceso**, **los avisos sonoros**, **la configuración de los interruptores**, **extracción de la información de localización** y, finalmente, **la configuración personalizada de los interruptores**.

El agente de **control de luces** es el ejemplo más claro de **adaptación** en el sentido clásico. Así, se encarga de modificar el estado de las luces en función del contexto. Apagando, por ejemplo, todas las lámparas cuando no quedan habitantes en el entorno, bajando la intensidad cuando se enciende la televisión o encendiendo la luz cuando se termina de ver ésta. A su vez incluye algunas reglas personalizadas como encender la luz de la mesa de Pablo cuando éste entra en la habitación o la de encima de la cafetera cuando el café está listo. Éstas últimas realizan más una función de aviso que de iluminación. Así, el entorno se flexibiliza, permitiendo usar dispositivos para tareas diferentes a las que presuponieron sus desarrolladores, de igual manera que lo haría una persona que, por las noches, prefiere la luz tenue de la televisión para iluminar su camino al baño.

Por su parte, el **control de accesos**, realiza una tarea que más bien podría ser denominada de **seguridad** que de adaptación. Dado que cada usuario tiene asociada una tarjeta RFID, este agente se encarga de abrir la puerta como reacción a la lectura de una de dichas tarjetas por el lector instalado en la puerta, algo así como una llave y su cerradura. Sabiendo que la puerta consta de picaporte para salir pero no para entrar y que el RFID se utiliza como identificador de entrada y de salida, el agente está configurado, sacando provecho del contexto, para que la puerta sólo se abra automáticamente cuando el usuario está intentando entrar, esto es, cuando no está ya dentro del entorno. Por otra parte cabe destacar que activar o desactivar llaves, dar de alta nuevas, hacer llaves maestras o asociar una tarjeta a otra persona es cuestión de segundos.

El tercer agente o de **avisos sonoros** tiene como función **comunicar** ciertos eventos o humanizar el entorno. Contando con los diferentes altavoces del sistema, varios de ellos con sintetizadores de voz instalados, este agente se encarga, entre otras cosas, de saludar y despedir a los distintos habitantes, lo cual sirve de feedback al resto de ellos –dado que el entorno está dividido visualmente, permite a "los del otro lado" saber quién acaba de llegar- así como notificar eventos especiales, como por ejemplo, "el café está listo".

En cuarto lugar tenemos el agente de **configuración de interruptores**. Este agente sirve como ejemplo para ilustrar la utilidad de tener un agente **asociado a cierto dispositivo**, una forma de encapsular su comportamiento de manera fácilmente configurable o adaptable. Por otra parte, haciendo uso de la información de contexto, se ha **aumentado la capacidad** del interruptor –único en una sala que consta de cinco luces, dos de ellas regulables- para que encienda y apague las luces paulatinamente a medida que es pulsado. Esta configuración transforma el significado clásico de "enciende la luz asociada" que tenía el interruptor al nuevo significado ambiental "quiero más luz". Dado que las luce se encienden y apagan –e incrementan y decrementan su valor, en el caso de las regulables- en el mismo orden, un usuario familiarizado con el entorno puede obtener cualquier configuración lumínica de la habitación mediante un procedimiento similar al de las *Torres de Hanoi*. Por otra parte, dado que la primera luz que se enciende y se apaga es la luz del techo –asociada originalmente al interruptor- un usuario no familiarizado con el entorno encenderá y apagará dicha luz, usando el interruptor como en cualquier casa, sin ser consciente de ningún mecanismo diferente.

Otra de las posibilidades del sistema de agentes queda recogida por el de **localización**. Quizá el más lejano al concepto de adaptación, se encarga de la **obtención de contexto** de alto nivel a partir de otro de más bajo nivel. En concreto este agente se encarga de inferir la información de localización de un buen número de personas en función de ciertas variables contextuales, tales como la lectura de tarjetas RFID o el estado de su computadora (encendida, apagada, con el salva pantallas). Este agente es un ejemplo de cómo utilizar el sistema de reglas no sólo para adaptar el entorno en función del contexto sino también para construir y digerir ese contexto para que sea usado por otras aplicaciones.

Finalmente el último agente o de **configuración personalizada de los interruptores** muestra una cara más avanzada de la personalización. Más concretamente este agente se encarga de cambiar el comportamiento del interruptor de manera que no maneje las luces sino que encienda y apague televisor y cafetera. Para ello hace uso de lo que se denominó **características de *meta-agente*** de manera que, mientras que una cierta tarjeta RFID está presente en el lector de RFID cercano el agente de interruptores se desactiva. Este hecho, combinado con que cada una de las reglas de este agente tiene como condición que dicha tarjeta este presente en el lector, provoca que el interruptor anule todos sus comportamientos pasados –gracias a que estaban encapsulados en un solo agente- para mostrar otros nuevos y personalizados. Este "detonador" (presencia de tarjeta en el lector) podría ser sustituido por cualquier otro como, por ejemplo, la presencia de cierta persona o de cierta otra, adaptando el entorno a sus habitantes

activando y desactivando agentes, algo así como si cada usuario viajase con su servicio tomando el control allí donde éste vaya.

Éstos agentes han ido creciendo poco a poco, a medida que los habitantes del entorno han ido demandando nuevos servicios. Su fácil y rápida implementación así como la flexibilidad que proporcionan los han convertido en sustitutos de diversas aplicaciones ad hoc que se encargaban previamente de estas tareas.

Agradecimientos

Este trabajo es parte del proyecto U-CAT que está financiado por el Ministerio de Ciencia y Tecnología con código TIN2004-03140.

Referencias

HAMILL, L. and HARPER, R. 2006. Talking Intelligence A Historical and Conceptual Exploration of Speech-based Human-machine Interaction in Smart Homes. International Symposium on Intelligent Environments (Microsoft Research, Cambridge, United Kingdom, Apr 5-7, 2006), 121-127

HAYA, P.A.; MONTORO, G. and ALAMÁN, X. 2004. A prototype of a context-based architecture for intelligent home environments. International Conference on Cooperative Information Systems (CoopIS 2004), Larnaca, Cyprus. October 25-29, 2004. 477-491.

HAYA, P.A. ; MONTORO, G. ; ALAMÁN, X. ; ESQUIVEL, A. and GARCÍA-HERRANZ, M. 2006. A mechanism for solving conflicts in Ambient Intelligent environments. Journal of Universal Computer Science, 12, 3, 284-296

HAYA, P.A. 2006. Tratamiento de la Información Contextual en Entornos Inteligentes. Escuela Politécnica Superior de la Universidad Autónoma de Madrid. Marzo de 2006

KIDD, C.K. ; ORR, R. and ABOWD, G.D. ; ATKENSON, C.G. ; ESSA, I.A. ; MACINTYRE, B. ; MYNATT, E. ; STARNER, T.E. and NEWSLETTER, W. 1999. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In Proceedings of the Second International Workshop on Cooperative Buildings - CoBuild'99"

MAES, P.; 1994. Agents that Reduce Work and Information Overload. Communications of the ACM, 7, 37, 31-40

WEISER, M. 1991. The computer for the 21st century. Scientific American, 265, 3, 94-104