UNIVERSIDAD DE DEUSTO

• • • • • • • • • • • • • • • • • •

Facultad de Ingeniería

Middleware framework para la configuración y personalización de entornos ubicuos por el usuario final

TESIS DOCTORAL PRESENTADA POR DON AITOR URIBARREN AXPE

PROGRAMA DE DOCTORADO: CIENCIA DE LA COMPUTACIÓN

DIRIGIDA POR

EL DR. DIEGO LZ. DE IPIÑA GLZ. DE ARTAZA

Y

LA DRA. ROSA IGLESIAS PÉREZ

El director	La codirectora	El doctorando
		

Bilbao, Abril de 2011

RESUMEN

Actualmente, la mayoría de los entornos inteligentes y ubicuos ofrecen unos determinados servicios al usuario de manera automática, como por ejemplo la televisión de muchos hoteles sirve para acceder a información variada, ofreciendo una guía de servicios, como búsquedas, selección de canales, despertador, etc. Sin embargo, como usuarios queremos muchas veces un entorno más inteligente y complejo, que involucre no sólo un único dispositivo sino un ecosistema variado y coordinado de dispositivos lo cual supone un gran desafío. La idea de que los mismos usuarios finales puedan educar el comportamiento del entorno resulta muy interesante, obteniendo servicios más complejos y adaptados hacia sus gustos.

La personalización que implica la adaptabilidad al contexto y configurabilidad de dispositivos heterogéneos, están siendo abordadas en la actualidad pero las soluciones existentes siguen siendo ad hoc. La adaptación y configuración de entornos ubicuos, fundamentada en el paradigma ECA (Event-Condition-Action) es otro aspecto básico a tener en cuenta.

Por otro lado, la provisión de mecanismos para la movilidad de aplicaciones que garantice su funcionalidad ante los continuos cambios de ubicación de los usuarios estudia la creación de aplicaciones tipo 'follow-me' que sean capaces de mover las configuraciones y mantener los estados de las mismas independientemente de la plataforma de ejecución y de los dispositivos existentes en cada ubicación. Las soluciones que se manejan, sin embargo, siguen sin cubrir las necesidades de los diferentes usuarios y entornos.

Esta tesis, propone un nuevo middleware *framework denominado CAHIM (Middleware* Configurable, Adaptable, Heterogéneo e Interoperable) que proporciona unos mecanismos adecuados para garantizar la interoperabilidad de las aplicaciones y de los dispositivos usando componentes y que posibilita la personalización y configuración de entornos ubicuos e inteligentes a través de las acciones definidas por el propio usuario. CAHIM incorpora un motor de reglas ECA que le dota de cierta inteligencia característica de los entornos ubicuos, facilitando el descubrimiento y asociación (*binding*) de servicios agnósticos a diversos protocolos y dispositivos heterogéneos subyacentes. Además, mediante un editor gráfico el usuario toma un papel central y coordina la personalización del comportamiento de su entorno. Para ello, se proporciona una nueva herramienta, TAMAmI (TAilor-Made Ambient Intelligence), con la que el usuario final no experto puede crear las reglas ECA que posteriormente son utilizadas por el motor de reglas.

ABSTRACT

Currently, most of intelligent or ubiquitous environments offer certain services to users in an automatic way, such as a television set in many hotels that allows accessing varied information, offering a guide to certain services, such as searches, selection of channels, alarm clock and so on. However, many times as users, we want a more intelligent and complex environment, involving not only a single device but also a varied ecosystem of devices, which poses a major challenge. The idea behind the fact that end users can educate/personalize an environment behavior is very interesting, getting services that are more sophisticated and are adapted to their pleasantness.

Personalization, which implies context-awareness and configurability of heterogeneous devices, is currently being addressed, although all proposed solutions are being ad hoc. Adaptation and configuration of ubiquitous environments based upon ECA (Event-Condition-Action) paradigm is another aspect to consider.

On the other hand, the provision of mechanisms to support application mobility, and thus to ensure application functionality despite changes in user location, addresses 'follow-me' applications. A follow-me application is one that moves configurations and keeps the state of the application regardless of execution platform and of existing devices at each user location. Nonetheless, the current solutions are far from covering needs from diverse users and environments.

This thesis proposes a new middleware framework, named CAHIM (Configurable, Adaptable, Heterogeneous and Interoperable Middleware) that provides mechanisms to support application and device interoperability and to enable configuration and personalization of ubiquitous environments. Personalization is achieved by means of actions defined by users themselves. CAHIM is provided with an ECA rule engine for supporting certain intelligence. Intelligence is a characteristic of ubiquitous environments that enables discovery and binding of agnostic services to diverse protocols and heterogeneous devices. In addition to this, users can play a key role and can coordinate personalization of their environment behavior through a graphical editor called TAMAmI (TAilor-Made Ambient Intelligence). It allows non-expert users to define their ECA rules, and the rule engine subsequently uses these.

AGRADECIMIENTOS

Quiero agradecer a los doctores Diego López de Ipiña y Rosa Iglesias la confianza que han depositado en mí, ya que gracias a ello he sido capaz de llevar a cabo este trabajo de investigación. Sus comentarios y aportaciones han sido imprescindibles para culminar esta tesis.

A la Universidad de Deusto y al centro tecnológico Ikerlan por poner a mi disposición los medios materiales y humanos necesarios. Agradecer también la colaboración y la ayuda de mis compañeros del área de Tecnologías de Software de Ikerlan, en especial a Garbiñe López, por los quebraderos de cabeza y aprietos soportados.

Tampoco me puedo olvidar de los de casa y familia en general. Gracias por la paciencia y todos los ánimos recibidos durante todo este tiempo.

Gracias a todos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE ACRÓNIMOS

.NET	Microsoft Framework API
AmI	Ambient Intelligence
API	Application Programming Interface
BDF	Bus Domótico Fagor
BML	Bean Markup Languaje
CAIPS	Context-Aware Information Push Service
CF	Compact Framework
CLIPS	C Language Integrated Production System
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CSS	Cascade Style Sheets
CSV	Comma Separated Values
DDC	Discoverable Device Controller
DLL	Dinamic Link Library
DLNA	Digital Living Network Alliance
DPWS	Device Profile for Web Services
DVD	Digital Versatile Disc
ECA	Event Condition Action
ER	Editor de Reglas
EXE	Executable
FTAM	File Transfer, Access and Management
GB	GigaByte
GENA	General Event Notification Architecture
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HAVI	Home Audio / Video Interoperability
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol

I/O	Input/ Output
IEC	International Engineering Consortium
IP	Internet Protocol
JESS	Java Expert System Shell
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LED	Light Emitting Diode
ME	Micro Edition
MR	Motor de Reglas
Mobi-D	Model-Based Interface Designer
MS Windows	Microsoft Windows
NFC	Near Field Communication
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OSGi	Open Services Gateway Initiative
OWL	Ontology Web Language
P2P	Peer-To-Peer
PC	Personal Computer
PDA	Personal Digital Assistant
PnP	Plug And Play
PROLOG	PROgramming in LOGic
QoS	Quality of Service
RAM	Ramdom Access memory
RFID	Radio-Frequency IDentification
RIA	Rich Internet Applications
SDK	Software Development Kit
SDP	Service Discovery Protocol
SEBIK	Sistema Eragile Bateratua IKerlan
SLP	Service Location Protocol
SOA	Service-Oriented Architecture
SVG	Scalable Vector Graphics

TAMAmI	TAilor-Made Ambient Intelligence
TCP	Transmission Control Protocol
TTS	Text To Speech
UDP	User Datagram Protocol
UIML	User Interface Markup Language
UMTS	Universal Mobile Telecommunications System
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
UUID	Universally Unique IDentifier
VB	Visual Basic
WAP	Wireless Application Protocol
WiFi	Wireless Fidelity
WS	Web Service
WSDL	Web Services Description Language
XAML	eXtensible Application Markup Language
XHTML	EXtensible HyperText Markup Language
XML	Extensible Markup Language
XSLT	eXtensible Stylesheet Language Transformation
XUL	XML based User Interface Language
XView	X window system based Visual/Integrated Environment for Workstations

Capítulo 1: Introducción

En este capítulo se describe la problemática existente en cuanto a la interoperabilidad, configuración y personalización en el marco de la Computación Ubicua o Inteligencia Ambiental. Se expone la motivación de este trabajo y sus objetivos y finalmente, se detalla la metodología utilizada a lo largo de toda la investigación, así como la estructura de la presente memoria.

1.1 Entornos ubicuos

La visión sobre la Computación Ubicua[1], término acuñado por Weiser, es aquella en la que los sistemas y dispositivos con capacidad de procesamiento se integran de manera transparente en la vida cotidiana de cualquier persona y colaboran entre sí [Weiser91]. Estos sistemas son capaces de responder a nuestras necesidades y condiciones proporcionando la información necesaria en cada momento y lugar. Esta visión comienza a hacerse realidad gracias a la evolución de las tecnologías sobre todo a nivel de miniaturización del hardware y de las comunicaciones inalámbricas. Hoy en día, disponemos en el mercado de un importante número de dispositivos como son los teléfonos móviles, las PDAs, los nodos sensores y actuadores para la automatización (por ejemplo, del hogar - Domótica), dispositivos que están o estarán embarcados (o embebidos) en nuestro entorno. Además, estos dispositivos no están aislados sino que son capaces de comunicarse con otros dispositivos e intercambiar información entre ellos, por ejemplo utilizando redes inalámbricas GPRS, UMTS, 3G, Bluetooth, Zigbee, entre otras, o a través de redes cableadas. Esta gama de nuevos dispositivos son capaces de procesar, guardar información y transmitirla a otros dispositivos.

Como no los vemos, no somos conscientes de su presencia, pero un creciente número de sistemas embarcados está ya en nuestra rutina diaria. Estos sistemas se encuentran entre otros en los microondas, reproductores de audio, coches, termostatos, cámaras de fotos, televisiones, semáforos, juguetes, sensores, actuadores, dispositivos electrónicos y otros muchos más. Esta cantidad de dispositivos de tamaño reducido, autónomos, con razonables capacidades de computación, comunicación cableada y/o inalámbrica [SmartHome09] junto con el desarrollo de múltiples aplicaciones están permitiendo que el propio entorno se comporte de acuerdo a las peticiones, preferencias o configuraciones establecidas por

[1] Este concepto es similar a otros términos en inglés como ‘pervasive computing’, ‘ambient intelligence’, ‘calm technology’, ‘internet of things’, ‘sentient computing’ y ‘everyware’ entre otros.

el usuario [Pronto09]. Es decir, en nuestro entorno están apareciendo nuevos dispositivos o sistemas embebidos que nos ofrecen nuevos servicios.

Actualmente, la mayoría de estos entornos inteligentes ofrecen unos determinados servicios primitivos al usuario, como el de encender las luces cuando alguien entra en una habitación o controlar el estado de encendido y apagado de un electrodoméstico u otro dispositivo. Sin embargo, muchas veces queremos un entorno más inteligente y complejo, que involucre y coordine un ecosistema variado de dispositivos. Por ejemplo, la acción de "Encender la alarma cuando no hay nadie en casa" implica más de un dispositivo (dispositivo alarma acústica y/o visual, varios sensores de presencia ubicados por toda la casa y dispositivo controlador). Al definir acciones en nuestro ambiente dada una determinada situación, estamos personalizando el comportamiento de nuestro entorno o ambiente. Tanto desde el punto de vista del programador como del usuario final, el poder definir el comportamiento de un entorno ubicuo es realmente retador siendo aún más difícil conseguirlo para todo tipo de usuarios. Para lograrlo, deben ser tenidos en cuenta varios aspectos, como son: la heterogeneidad e interoperabilidad de dispositivos y servicios, la adaptabilidad y configurabilidad de los entornos, así como la personalización por parte del usuario final. Con la personalización de un entorno, los usuarios podrán ser capaces de acceder y manipular la información y servicios relacionados por ejemplo con su localización o con la hora del día.

1.1.1 Heterogeneidad

Para que nuestros entornos sean cada vez más inteligentes, se necesita conseguir un acceso transparente *(seamless*) a cualquier servicio disponible, por ejemplo, desde una aplicación que requiera el uso de unos servicios dados por diversos dispositivos en la red o bien desde un dispositivo que usará los servicios de otros dispositivos. Uno de los retos que aparece en este dominio es el de abordar la heterogeneidad de los dispositivos y de los servicios de las aplicaciones.

Actualmente, existen en el mercado multitud de dispositivos comerciales y de uso extendido con capacidades de comunicación, tales como PCs, PDAs o teléfonos móviles, que pueden ejecutar programas desarrollados en diferentes lenguajes y con distintos *middlewares*, pero que en muchos casos no se comunican con otros dispositivos, como pueden ser los dispositivos domóticos [Jimeno+04]: actuadores, sensores, electrodomésticos, etcétera. El problema reside en la diversidad de software de cada uno de los dispositivos y los protocolos de comunicación propietarios, que dificultan la comunicación entre los mismos [Uribarren+07a]. Es aquí donde aparecen nuevos problemas, como son lograr la accesibilidad, interoperabilidad y portabilidad. Dotar con capacidades Plug&Play a los dispositivos (conocida también por PnP) se convierte en una tarea imprescindible en este aspecto ya que las restricciones de hardware y software pueden llegar a ser insuperables en la mayoría de los casos [Thomson+07].

1.1.2 Interoperabilidad de servicios y aplicaciones mediadas por el contexto

La interoperabilidad puede ser definida como la capacidad de dos o más sistemas para intercambiar información y usar la información que se ha intercambiado, por lo tanto, dichos sistemas puedan trabajar conjuntamente en la consecución de un fin común [IEEE91]. Por ejemplo, un entorno inteligente debe ser consciente de la presencia de usuarios (sensible a la localización del usuario), sensible al contexto (captación de la información del ambiente) y sensible a los deseos, hábitos, acciones y emociones de los mismos [Salvador+05, Saha+03]. Para satisfacer estos requisitos, las aplicaciones/dispositivos deberían tener acceso transparente y ubicuo a los elementos del sistema o sistemas y cooperar entre ellos para ser conscientes del entorno que le rodea y proveer inteligencia e interacción transparente hacia el usuario [Ferscha03]. En este tipo de escenarios, los diferentes nodos o elementos de los sistemas se refieren a los servicios disponibles en un entorno. El *middleware framework* subyacente en todo sistema ubicuo debería proveer los mecanismos para que esta comunicación y cooperación sean posibles. Además, el *middleware* debería también facilitar a las aplicaciones tareas como la gestión de protocolos de transporte y gestión de fallos de comunicación [Saha+03, DaCosta+08].

A día de hoy, la tecnología parece estar lo suficientemente madura para ofrecer y crear entornos inteligentes, pero lo cierto es que la realidad es bien distinta. La mayoría de los dispositivos siguen estando aislados, comunicándose solamente con aplicaciones propietarias, aislados sin comunicación con otros dispositivos, esto es, no colaborando al servicio de los objetivos definidos en las aplicaciones [Aarts+03, Uribarren+07b].

1.1.3 Movilidad

Frecuentemente, la personalización de un entorno puede llevar a tomar decisiones como pueden ser la activación, desactivación y movilidad de servicios y componentes (es decir, gestionar el ciclo de vida de los servicios). Un ejemplo claro surge al pensar en el gestor (aplicación ubicua) para mover aplicaciones o servicios asociados a una nueva posición del usuario: escenarios ‘follow-me’. Para ello, el *middleware framework* debería proveer de los mecanismos necesarios para la instalación y ejecución remota de aplicaciones en distintos dispositivos. Es decir, el objetivo sería ofrecer al usuario la misma funcionalidad cualquiera que sea su ubicación, dispositivo en uso, u otra serie de parámetros del entorno que en un momento dado se estén utilizando; todo de forma autónoma, transparente y ubicua para el usuario, instalando y ejecutando aplicaciones dependientes de cada *framework* a demanda, siendo independientes los parámetros de configuración del código ejecutable.

Actualmente, existen en el mercado multitud de dispositivos de computación, como PCs, PDAs, teléfonos móviles que pueden operar a través de redes de comunicación diferentes y ejecutar diferentes *frameworks* no compatibles por tener diversas capacidades de procesamiento, memoria, etcétera. La adaptación de las aplicaciones

para que puedan ser ejecutadas en cualquier dispositivo es una tarea costosa y complicada en la mayoría de los casos, ya que las restricciones de hardware y software suelen llegar a ser insuperables. Esto ha sido objeto de muchos trabajos de investigación [Uribarren+05, Parra+06]. Sin embargo, la migración de servicios entre dispositivos que comparten diferentes plataformas sigue siendo una demanda no satisfecha. Además, estas cuestiones introducen y enfatizan el carácter dinámico necesario que una aplicación debería tener a la hora de formar parte de una u otra infraestructura de manera transparente; este tipo de capacidad o mecanismo resulta esencial en entornos de computación ubicua para soportar aplicaciones sensibles al contexto, que necesiten intercambiar información adaptándola entre los diferentes dispositivos y reduciendo la interacción entre persona y aplicación [Dey01].

1.1.4 Adaptabilidad y configuración

La gestión de la información, como es el estado de los servicios y los datos provenientes de los diferentes sensores y dispositivos heterogéneos móviles, se convierte en una tarea laboriosa para el programador o usuario final. Los entornos ubicuos son muy dinámicos y se necesitan mecanismos para la gestión de los dispositivos y servicios que aparecen y desaparecen y de otros cambios en el entorno (contexto). El *middleware framework* debería proveer los mecanismos necesarios para contemplar este comportamiento dinámico del entorno. La adaptabilidad o flexibilidad es la capacidad de un sistema o componente para modificar su comportamiento y ser utilizado en aplicaciones o entornos diferentes para los que fue específicamente diseñado [IEEE00].

En este sentido, las aplicaciones ubicuas necesitan ser conscientes de los cambios en el entorno y de las necesidades que el usuario tiene, y ante ello adaptarse e incluso, muchas veces, anticiparse a ellos. Ser consciente del contexto que rodea al usuario es de vital importancia para proveer al usuario de servicios adaptados a su entorno. Para ello es necesario dotar al *middleware* de mecanismos que posean un procesamiento inteligente de la información recibida, y de servicios confiables.

Un *middleware framework* debería encargarse de facilitar la comunicación entre dispositivos heterogéneos, de recolectar la información de contexto y de los servicios disponibles en los dispositivos, y finalmente de procesar toda la información generando nueva información, resultando todo ello transparente para las aplicaciones. Además, sería esencial adoptar estrategias que se adaptasen a los continuos cambios del entorno y a las necesidades o preferencias del usuario. En este sentido, sería interesante que el *middleware* propuesto fuese un paso más adelante, ofreciendo mecanismos para almacenar información referida al contexto y generar nuevos servicios a partir de ellos

1.1.5 Personalización del entorno

Aunque el término configuración está directamente relacionado con cómo “se percibe el entorno y se actúa según los cambios producidos”, en esta tesis vamos a usar el

término personalización para referirnos a cambiar el comportamiento o funcionalidad de un entorno. Las personas queremos que nuestro entorno se comporte según nuestras preferencias o que se ajuste a ciertas condiciones, por ejemplo, queremos controlar el comportamiento de un dispositivo en un contexto determinado o bien que ocurra algo cuando estamos en una habitación. Para poder personalizar un determinado entorno, independientemente del nivel de conocimientos informáticos del usuario final, resultaría deseable la inclusión de una interfaz sencilla y de fácil manejo

1.2 Motivación de la tesis

Los avances logrados en la miniaturización y compactación de dispositivos electrónicos han permitido embarcar microprocesadores prácticamente en cualquier dispositivo y lugar. Debido a ello, hoy en día se dispone de dispositivos de tamaño reducido con unas capacidades de computación y comunicación impensables hace unos años que pueden ser utilizados en entornos no solo industriales. Este tipo de dispositivos inteligentes con un costo asumible por el usuario final nos ofrecen nuevos servicios que se están introduciendo en nuestros hogares.

Por otro lado, la personalización de un entorno involucra el estudio, análisis y búsqueda de soluciones para lograr la interoperabilidad de dispositivos heterogéneos, la movilidad, la adaptabilidad y la configurabilidad; retos que han sido y están siendo abordados en la actualidad por diversos grupos de investigación, pero cuyas soluciones siguen siendo ad hoc.

Existen diversas propuestas de *middleware framework* que tratan de ofrecer mecanismos de interoperabilidad entre dispositivos y aplicaciones; sin embargo, en su mayoría, aunque logran cierta interoperabilidad, no solucionan todos los desafíos existentes a la hora de personalizar un entorno. Siguen siendo islas de funcionalidad que ofrecen servicios agnósticos a los usuarios. Los dispositivos deben poder comunicarse e intercambiar información independientemente de las APIs (*Application Programming Interface*) de los dispositivos, protocolos o *framework* de ejecución. Además, el anuncio y descubrimiento de los servicios ofrecidos por los dispositivos deberían ser transparentes para cada aplicación e independientes del protocolo de anuncio y descubrimiento usado (por ejemplo, OSGI, UPnP, Web Service o HAVi).

Además, en este tipo de entornos se busca que el usuario no perciba la tecnología que le rodea y que su intervención sea mínima, por lo tanto, lo ideal sería conseguir que los servicios se adapten al entorno, que se mantengan vivos incluso cuando la gente se mueva de una ubicación a otra, y que no necesiten de mantenimiento -‘zero administration’- [López-de-Ipiña+06]. La idea de que los mismos usuarios puedan personalizar el comportamiento de su entorno, obteniendo servicios más complejos y adaptados a sus gustos, resulta muy interesante. Para realizar esta labor de personalización del entorno, sería deseable contar con una herramienta gráfica amigable y fácil de usar para cualquier usuario.

1.3 Objetivos de la tesis

El objetivo general de este trabajo de investigación es dar una solución adecuada a toda esta problemática; se centra en la creación de un *middleware framework* que exhiba las siguientes características:

- **Heterogeneidad:** Ser compatible con gran variedad de dispositivos y servicios.
- **Interoperabilidad:** Incluir los mecanismos necesarios para facilitar la comunicación y el trabajo en grupo de una red heterogénea de dispositivos y servicios.
- **Adaptabilidad/Dinamismo:** Tener la capacidad de detectar y gestionar los cambios en el entorno (contexto). Por ejemplo, detectando nuevos dispositivos y/o servicios, errores o cambios de localización del usuario, aunque también se cree que cierta gestión de adaptabilidad debería ser llevada a cabo por las propias aplicaciones.
- **Movilidad –'follow-me':** Poder mover los servicios y contemplar los mecanismos necesarios para reconfigurar y mantener el estado de las aplicaciones después de modificar su ubicación.
- **Configurabilidad:** Ser capaz de modificar las configuraciones de las aplicaciones en función de los servicios o recursos de red disponibles.
- **Personalización:** Proporcionar una herramienta de configuración o editor gráfico para que el usuario final puede personalizar el comportamiento de su entorno.

Para lograrlo, planteamos la realización de las siguientes tareas:

- Establecer los requisitos de un *middleware framework* para lograr la interoperabilidad de los distintos dispositivos, de tal forma que puedan trabajar juntos a pesar de su diversidad (heterogeneidad), y cuyos servicios ofrecidos puedan ser usados fácilmente por cualquier aplicación.
- Estudiar y analizar las tecnologías de dinamismo de los entornos ubicuos (recursos que aparecen y desaparecen, configuraciones que cambian en función de los servicios disponibles) para procesar la información del entorno generando a su vez, nueva información del contexto y de los servicios disponibles.
- Analizar en profundidad los mecanismos ofrecidos en la computación móvil para lograr la movilidad de los servicios ofrecidos por ciertas aplicaciones en función de la ubicación del usuario, así como la reconfiguración de aplicaciones entre dispositivos con características diferentes.
- Diseñar e implementar un nuevo *middleware framework* que se ajuste a los requisitos fijados para lograr la personalización y automatización del entorno.
- Desarrollar una interfaz amigable para que el usuario final pueda configurar ambientes y personalizar su entorno definiendo sus preferencias

- Evaluar el comportamiento del *middleware framework,* su rendimiento en diferentes situaciones y la usabilidad del editor gráfico por parte de los usuarios finales.

1.4 Metodología de investigación

La metodología de investigación seguida se muestra de forma gráfica en la figura 1.1. Los pasos seguidos obedecen a la metodología de investigación científico-técnica que en resumen son:

- Revisión estado del arte
- Detección de la problemática
- Fijar hipótesis y objetivos: Requisitos de la solución
- Desarrollo
- Experimentación/Validación
- Conclusiones
- Difusión

Esta investigación se traduce en las tareas siguientes:

1. Revisión del estado de arte –publicaciones, herramientas, trabajos de investigación, etc.- en el ámbito de la Inteligencia Ambiental, de la provisión de entornos ubicuos, y de la movilidad de aplicaciones.
2. Análisis crítico (alcances, limitaciones, ventajas y desventajas) de los mecanismos que integran el manejo de eventos y la toma de decisiones para facilitar el desarrollo de aplicaciones sensibles al contexto, y por tanto aplicaciones con cierta inteligencia.
3. Establecimiento de requisitos de un nuevo sistema que solvente los problemas existen en el desarrollo de servicios y/o aplicaciones interoperables y móviles en entornos domésticos u ofimáticos, para apoyar la movilidad humana por medio de un contexto personalizable.
4. Diseño de los diferentes componentes de un *middleware framework* denominado CAHIM basado en sistemas telemáticos e implementado como un conjunto de módulos independientes con capacidad de interacción.
5. Evaluación del rendimiento y usabilidad en laboratorio y en entornos reales, del prototipo que implementa el *middleware framework* propuesto y de sus componentes en cada fase del ciclo de vida mediante una herramienta llamada TAMAmI (*TAilor-Made Ambient Intelligence*).
6. Establecimiento de conclusiones y presentación de los resultados y conclusiones a la comunidad científica para su consideración y aprobación.

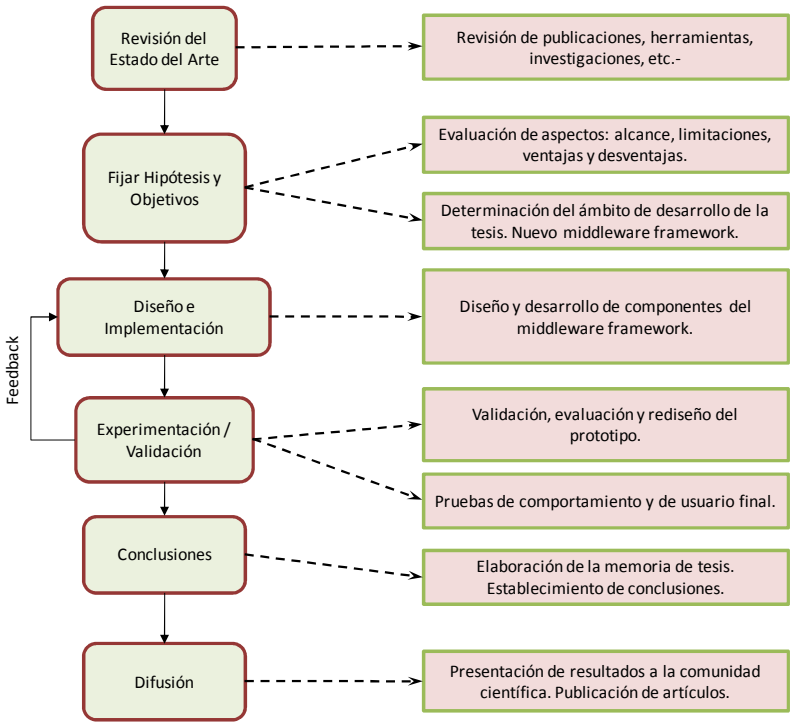


Fig. 1.1 Metodología de Investigación.

1.5 Contenido de la memoria

Tras el presente capítulo de introducción, se presenta el estudio del estado del arte realizado. En los capítulos 3, 4, y 5 se describen los diferentes componentes y capas del *middleware framework* que a modo de resumen se introducen y detallan a continuación.

En el capítulo 3 se presenta un estudio sobre la interoperabilidad en cuanto a los nuevos dispositivos disponibles en el mercado y la problemática asociada a los mismos. Se analizan los requisitos de los dispositivos y las diferentes familias de protocolos de descubrimiento de servicios para garantizar la interoperabilidad necesaria de los escenarios planteados y así dar soporte a la adaptación requerida. A continuación, se propone una serie de capas intermedias para el intercambio de información entre aplicaciones y dispositivos (figura 1.2) que serán las responsables de la adecuación y conversión de los datos transferidos entre los dispositivos y resto de componentes del *middleware framework*.

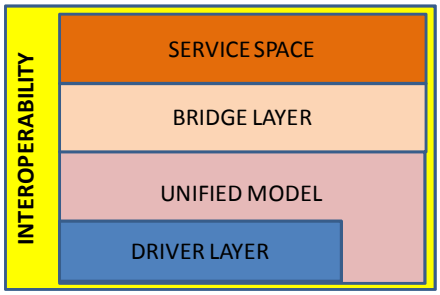


Fig. 1.2 Vista componente Interoperabilidad del middleware framework

En el capítulo 4 se describe qué se entiende por sensibilidad al contexto y lo que supone en cuanto a la movilidad de aplicaciones respecto a la posición del usuario en un entorno ubicuo. El componente de movilidad mostrado en la figura 1.3 se analizada en este capítulo. Se describe la arquitectura que facilita dicha movilidad de aplicaciones, escenarios conocidos como 'follow me', ilustrándolo con ejemplos prácticos, y se analiza cómo debería adaptarse el sistema a las necesidades y preferencias establecidas por el usuario configurando el entorno circundante.

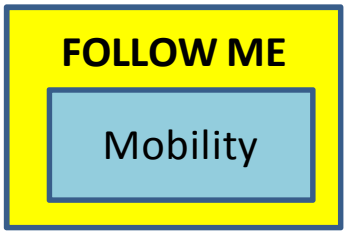


Fig. 1.3 Vista componente 'Follow-Me' del middleware framework

En el capítulo 5, tras exponer el estado del arte de la personalización dentro del ámbito de los entornos ubicuos, el paradigma ECA y su aplicación en la personalización de los entornos ubicuos, se establecen los aspectos relativos a la inteligencia necesaria para el ámbito de aplicaciones ubicuas a las que se pretende abarcar (figura 1.4) y se explica la solución aportada para la creación del motor de reglas ECA que satisface los requisitos planteados.

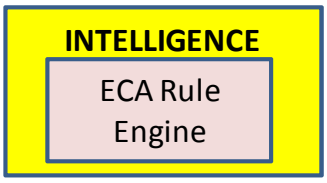


Fig. 1.4 Vista componente Inteligencia del middleware framework

En el capítulo 6 se analiza el estado de arte en cuanto a los configuradores gráficos personales en el ámbito de la computación ubicua, introduciendo el paradigma del 'end user programming', justificando la utilización de un editor de reglas adaptado para cubrir las necesidades de personalización de entornos ubicuos. Este editor se denomina TAMAmI (*TAilor-Made Ambient Intelligence*) (Fig. 1.5).

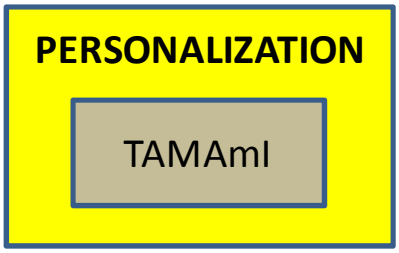


Fig. 1.5 Vista componente TAMAmI de la aplicación de interacción del usuario

En el capítulo 7, se presenta la evaluación del sistema configurador expuesto en el capítulo 5 y la validación de usuarios efectuada sobre la herramienta gráfica TAMAmI propuesta para generación de reglas personales.

En la figura 1.6 se muestra gráficamente, a modo de resumen, los diferentes elementos del *middleware framework* desarrollado para entornos inteligentes.

Finalmente, el capítulo 8 presenta las conclusiones haciendo hincapié en las aportaciones originales de esta tesis.

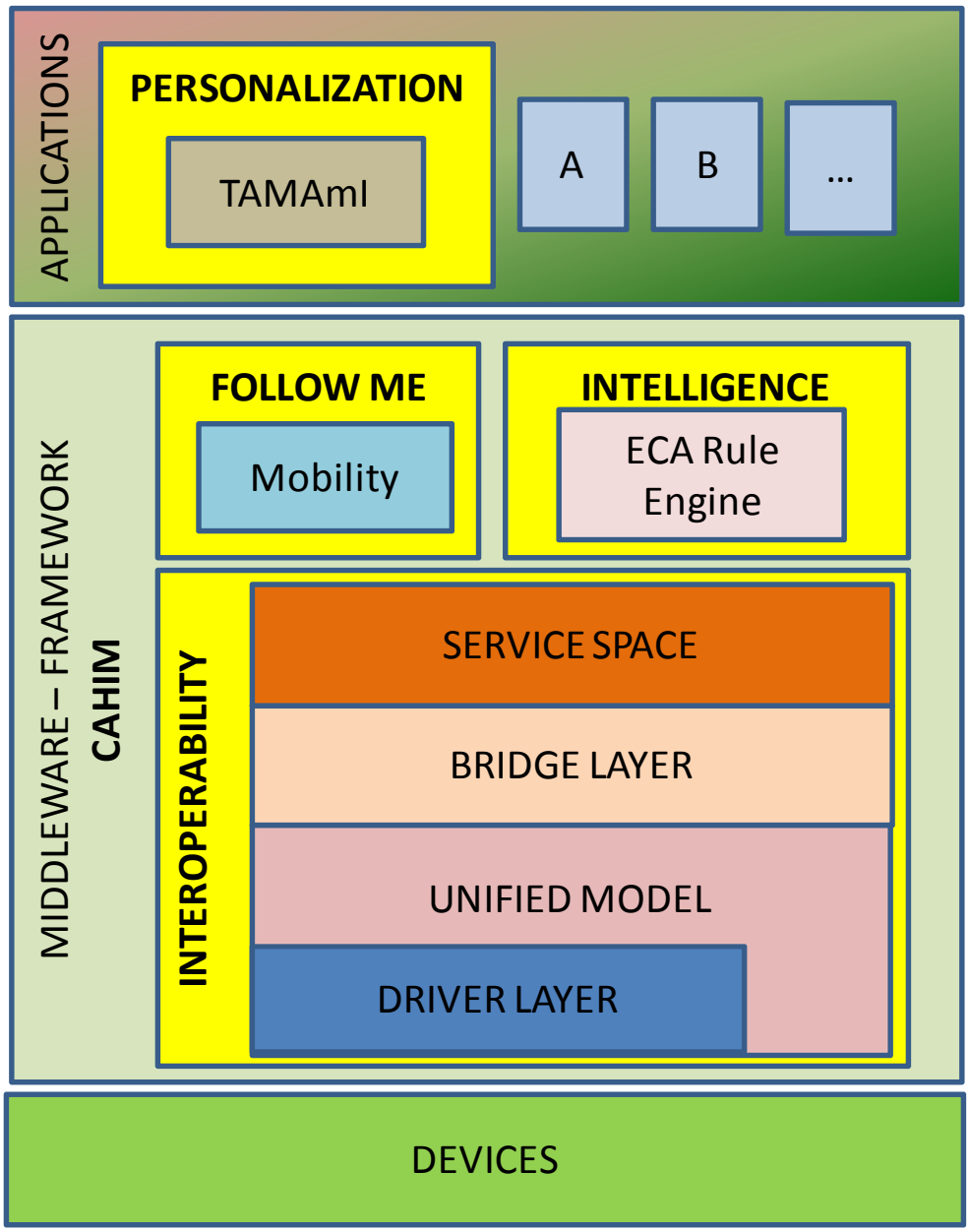


Fig. 1.6 Vista middleware framework para entornos inteligentes

Capítulo 2: Estado del arte

En este capítulo se recoge el estado del arte de cada uno de los componentes del middleware framework mencionados en el capítulo de introducción: interoperabilidad de aplicaciones y dispositivos, movilidad de aplicaciones, configuración y personalización de entornos ubicuos.

2.1. Interoperabilidad

En la computación ubicua, a la vez que los dispositivos son capaces de ofrecer más servicios de todo tipo, se verá un aumento en el número de los mismos [Weiser91]. Surgen dos problemas con esta gran densidad de servicios disponibles. El primer problema es la posibilidad de que exista una gran cantidad de servicios similares o solapados, por lo que los servicios han de proveer la suficiente información para diferenciarse unos de otros. El segundo problema es la selección de un servicio en el momento de su ejecución, la selección del más cercano, más estable... en otras palabras, más adecuado en cada momento. Por lo que, cuanta más información sea ofrecida por los servicios, más fáciles serán de organizar y utilizar.

Para facilitar el desarrollo de aplicaciones que utilizan servicios y reducir la complejidad de su administración, se ha desarrollado una serie de protocolos de descubrimiento de servicios (Service Discovery Protocol - SDP). UPnP [UPnP10], Jini [Jini10], SLP [SLP10] o DLNA [DLNA10] son ejemplos de protocolos de uso muy extendido. Sin embargo, esta diversidad significa que existen espacios de servicios diferenciados, cada uno de ellos representado con un formato particular. Por ello, se necesiten puentes o mapeos entre estos protocolos de descubrimiento de servicios de manera que las aplicaciones puedan acceder a cada uno de ellos de forma transparente [Salvador+05]. Existe una dificultad añadida, no sólo desde el punto de vista del usuario final de la aplicación sino que de los programadores también; es necesario conocer de manera unívoca la forma en la que el servicio ha sido publicado para poder instanciarlo. Por otra parte, compartir información suficiente entre los dispositivos puede garantizar una tasa de fallo baja como por ejemplo se demuestra en [Hightower +01] para localizar objetos en un entorno. Destacar que nuestro *middleware* incluye puentes entre protocolos, como se explicarán detalladamente más adelante. De esta manera, las aplicaciones finales o de alto nivel podrán utilizar cualquier.

Trabajos como [Coulson+02] se centran en adaptar las aplicaciones a los dispositivos o recursos cambiantes. Mediante el uso de mecanismos de reflexión, logran configurar en el momento de la primera instalación o despliegue el sistema en cuestión cargando y ejecutando los módulos y las configuraciones deseadas. Después en tiempo de ejecución utilizan el mismo mecanismo para reconfigurar los sistemas. Aunque en el

trabajo [Grace+03] proponen una forma de proveer información de los recursos de manera fija, utilizando WSDL, la abstracción completa para las aplicaciones se debería lograr sin forzar que las aplicaciones tengan que ser adaptadas a un mecanismo de descubrimiento SDP y forzadas a la utilización del mismo. Otras estrategias para la generación de sistemas de descubrimiento de servicios y que tienen en cuenta la tolerancia a fallos, rendimiento, escalabilidad, interoperabilidad con otros mecanismos de descubrimiento, la independencia de plataformas, la estandarización y la posible implementación de lo presentado para la adaptación de aplicaciones se analizan en [Liman+05]. Su importancia radica en que proveen pistas de la información que es relevante y los mecanismos, basados en servicios web, que utilizan para garantizar componentes débilmente acoplados que proveen una escalabilidad adecuada. No obstante, aunque a nivel de protocolo la interoperabilidad está cubierta, no así para los dispositivos heterogéneos ni para los distintos modos de acceso de los mismos, es decir, la inclusión de cualquier tipo de dispositivo en el sistema y su acople con los protocolos existentes no está cubierta.

El proyecto MobileMan [Conti+04] define una arquitectura en capas para el intercambio de información, muy similar a como lo hace un *stack* clásico de protocolos (red, transporte, aplicación...). Utilizan un elemento vertical denominado 'Network Status' para mantener informada a cada capa con el estado de las otras capas, con el objetivo de mejorar el dinamismo y rendimiento en redes móviles *ad hoc* dinámicas. Y aunque el ámbito de aplicación no sea el mismo, el protocolo de comunicación utilizado entre todas las capas es un mecanismo interesante. Nuestra propuesta incorpora mecanismos para el paso de información entre capas, haciendo hincapié en la adaptabilidad en la capa 'bridge', tal y como se verá más adelante en el apartado 3.3.

Existe otra serie de sistemas como Ninja [Gribble+00], Gaia [Roman+02] y Plan B [Ballesteros+05] basados en la utilización de *middleware* específicos para implementar y distribuir servicios.

Ninja se centra en la creación y composición de servicios de forma dinámica, configurable y escalable [Gribble+00]. Aunque pueda ser un buen sistema para compartir o enviar información para posteriormente adaptarla a los distintos dispositivos, tal y como se abordará en el capítulo 4, la necesidad de crear para cada dispositivo 'active proxies' tal y como ellos lo denominan, para compartir información y mantener el estado con una 'base' o sitio centralizado, requiere una infraestructura siempre conectada y debidamente actualizada, que en sí misma resulta complicada de gestionar por cualquier aplicación.

Gaia es básicamente un sistema operativo que trabaja con lo que ellos denominan espacios físicos (Physical Spaces -PS) [Roman+02]. Estos espacios físicos se pueden asociar con las estancias por las que un usuario va pasando en el hogar por ejemplo. Así, un dispositivo entra y sale, o se activa y desactiva de los diferentes espacios físicos en función de las preferencias del usuario o del estado a recrear en cada momento. Las funciones básicas para la intercomunicación de los dispositivos en los

diferentes espacios físicos, como si de un sistema operativo se tratara, como son el paso de eventos, sistema de archivos, seguridad, procesos... las implementan, aparte de incluir soporte para el contexto, la localización, movilidad de dispositivos... Las aplicaciones han de ser generadas de forma muy genérica, sin tener en cuenta el hardware sobre en el que van a ser ejecutadas. De esta forma, las aplicaciones pueden ser distribuidas para cada PS de forma automática. La interacción con otro tipo de *middlewares* existentes, es decir, la interoperabilidad que debería haber con otros sistemas, se puede decir que a nivel de *middleware* no existe, ya que todo lo que no sea PS no puede interactuar con el sistema.

Plan B es otro sistema operativo diseñado para trabajar en entornos distribuidos, donde una serie de recursos disponibles intercambian información variada [Ballesteros+05]. Plan B se centra en copiar o mover archivos de configuración desde lo que ellos denominan una caja o‘box’ a otra. Las aplicaciones pertenecientes a una caja se autoconfiguran con la información que se envía de una caja a otra, por lo que el intercambio de información solo se limita a los datos de un programa. La información se comparte entre aplicaciones a alto nivel.

Por otro lado, las especificaciones OSGi (*Open Services Gateway Initiative)* definen un entorno de desarrollo y ejecución para los servicios definidos como componentes. Los servicios han de ser especificados utilizando interfaces Java [OSGI]. Las aplicaciones OSGi, denominadas ‘bundles’, pueden implementar estas interfaces y registrarlas utilizando el servicio del *middleware* OSGi denominado ‘Registry’. El *middleware* hace de intermediario entre los distintos bundles, de manera que lo que un *bundle* ofrece otros bundles lo reutilizan. Esta es la mayor limitación del *middleware* OSGi. Aunque la interoperabilidad está garantizada entre los distintos bundles sobre la misma máquina, si tenemos un sistema distribuido con distintos dispositivos tipo PDA, Smartphone... los mecanismos ofrecidos por el *middleware* no garantizan la correcta ejecución de los distintos *bundles* sobre las PDAs y *smartphones*, entre otros. La razón es que si falla una dependencia entre *bundles*, falla la instalación de la aplicación correspondiente. Es decir, los *bundles* aunque funcionan de manera independiente, contienen enlaces que han de ser direccionados y localizados en el momento de la instalación para su correcta configuración. Otra limitación de OSGi es que dentro del propio *middleware* la dependencia entre *bundles* puede llegar a ser crítica en un entorno distribuido, de tal forma que hay que garantizar el correcto funcionamiento de todos los *bundles* para garantizar el resultado final deseado. Por el contrario, los mecanismos de carga/descarga, instalación/desinstalación de *bundles* son perfectamente válidos y deseables en un entorno ubicuo, a pesar de que cuando se inician los *bundles*, su inicialización deba ser en un orden preestablecido debido a la dependencia entre los mismos.

2.2 Movilidad

Con la proliferación de todo tipo de dispositivos de tamaño reducido con capacidad de conexión, ejecución de programas variados, acceso remoto, entre otras características, surgen otra serie de problemas y necesidades. Los problemas específicos de las aplicaciones distribuidas se convierten en críticos en entornos en los que los usuarios utilizan dispositivos heterogéneos. Los datos y elementos compartidos entre los dispositivos deben ser adaptados, presentados e intercambiados desde diferentes puntos de vista, en función del usuario, programa o sistema operativo utilizado en cada dispositivo, capacidad de procesamiento y almacenamiento [Weiser91] [Uribarren+06a], y siempre de manera segura y transparente para el usuario final. En este sentido, la interoperabilidad de/entre los dispositivos y aplicaciones juega un papel crucial a la hora de dar el servicio deseado al usuario final. Este componente del middleware pretende dar solución a la problemática del movimiento o traslado de las aplicaciones y/o servicios reaccionando frente a los cambios de ubicación del usuario. Este aspecto de la movilidad de aplicaciones se ha abordado desde diferentes puntos de vista a lo largo del estado de arte y es lo que en los siguientes apartados se analiza.

2.2.1 Trabajos basados en la adaptabilidad

Bean Markup Languaje (BML) es un lenguaje basado en XML para la configuración de componentes JavaBeans [BML10] [JavaBeans10]. El lenguaje es directamente ejecutable, es decir, el resultado de interpretar un script tipo BML es el ejecutarse las instrucciones indicadas en el script. Aunque desde el punto de vista funcional puede servir para realizar ciertas tareas, el hecho de que solamente se pueda utilizar Java es un impedimento.

DACIA es un *framework* que proporciona mecanismos para crear aplicaciones groupware que tratan de adaptarse a los recursos existentes en un dispositivo, y soporta la movilidad de usuarios y reconfiguración dinámica [DACIA10]. En el momento de la ejecución, los distintos componentes pueden ser instalados/desinstalados o movidos de un dispositivo a otro, sin que haya pérdida de conectividad con otros usuarios y programas. La estructura de la aplicación no varía y el flujo de datos en el sistema no es alterado. Reduciendo la dependencia de ciertos hosts gracias a la movilidad de componentes, las aplicaciones se hacen más robustas y más tolerantes a fallos de hosts y caídas de conexión. Pero, por otro lado, no se define cómo realizar la adaptación a las características propias de cada dispositivo, es decir, una aplicación en PC que use un entorno gráfico complejo no puede ser dinámicamente migrada a una PDA con unos recursos gráficos más limitados. Aunque la aproximación de DACIA puede ser la adecuada en muchos aspectos, podemos decir que falla en aspectos como la adaptación dinámica del propio contexto del usuario.

En el apartado anterior se han citado los siguientes trabajos [Ballesteros+05, Roma+02], en los que se aborda la reconfiguración de aplicaciones. Su principal

aportación es la de proveer mecanismos para guardar y restablecer las configuraciones y parámetros de ejecución entre diferentes espacios y ubicaciones en los que se ejecutan las aplicaciones cuando un usuario se mueve. En cambio, el *middleware* propuesto pretende ir más allá, no sólo moviendo las configuraciones y preferencias de los usuarios, sino que moviendo todo lo referente a los programas involucrados, instalando y desinstalando si fuera necesario, tal y como se explica en el capítulo 4.

Por otro lado, en [Becker+04a] se describe un sistema dirigido a componentes PECOM. En la misma se muestra una arquitectura no centralizada para la composición dinámica de servicios por parte de las aplicaciones, de tal manera que las aplicaciones detectan y descubren qué servicios se encuentran en su entorno para poder utilizarlos. Por ejemplo, una aplicación de monitorización de la temperatura, cada vez que el usuario se mueve con su PDA por las diferentes estancias de la casa, la PDA detectará los diferentes termostatos y en cada momento mostrará la temperatura de cada uno de ellos. Aunque este tipo de adaptación y reconfiguración de aplicaciones es interesante, la adaptación a las necesidades del usuario se pretende que sea total, enfocado a no perder y seguir ofreciendo la misma funcionalidad cualquiera que sea su ubicación. Es decir, la composición dinámica de servicios no va a ser suficiente para seguir proveyendo de la misma funcionalidad al usuario, por lo que a continuación se propone una arquitectura con los mecanismos asociados para facilitar la movilidad de aplicaciones y configuraciones y/o preferencias de los usuarios en sus desplazamientos entre diferentes estancias de un hogar o incluso entre un puesto de trabajo en la oficina y el ordenador de casa.

Para el caso en el que un usuario está escuchando música y leyendo su periódico preferido on-line en una ubicación y se mueve de sitio, la música y el periódico automáticamente se mueven con él, a la vez que se adaptan otra serie de recursos existentes a su alrededor. El estado actual del usuario es archivado y cuando haga falta reproducirlo en cualquier otro momento en cualquier otro lugar, no hay más que enviar el fichero de su configuración, posibilitando que el sistema pueda realizar un ‘Suspend&Resume’, parecido a lo que proponen [Kozuch+02] [Stefanelli+08], de manera que el trasvase de aplicaciones y configuraciones sea transparente para el usuario final.

2.2.2 Trabajos basados en la generación de interfaces de usuario

XWeb propone una metodología para definir interfaces de usuario utilizando el lenguaje XView [Satyanarayanan04]. XView especifica una serie de objetos de interfaz estándar denominados ‘interactors’ que han de ser implementados para cada plataforma destino en la que queramos utilizar aplicaciones XWeb. Básicamente, los ‘interactors’ se tratan de cadenas de texto representando todo tipo de información (incluyendo la representación de números). El contenido de la página HTML está claramente separado de los parámetros de formato y diseño de la página, esto quiere decir que las páginas creadas con este método no disponen de ningún tipo de

información relativa al aspecto ni de cómo deberían de ser representadas. Ambas informaciones (contenido y continente) son almacenadas en ficheros diferentes que son unidas en el momento de compilación.

XWeb es un programa en Java que utiliza tecnologías web XML, XSLT, CSS y SVG para crear sitios en internet que se pueden dejar disponibles en cualquier servidor para su posterior uso. XWeb puede utilizar cualquier tipo de entrada en formato XML, siendo XHTML el formato más simple y pudiendo mezclar formatos XML personales o cualquier XML que se tenga. Esto es posible debido a los mecanismos de compilación que convierten ficheros XML en páginas HTML e incluso objetos adicionales como imágenes, botones, 'banners' (forma de publicidad en Internet), etc. Una vez que la página ha sido generada, se puede mostrar en cualquier navegador utilizando cualquier servidor web.

El lenguaje de interfaces de usuario XUL basado en XML pretende facilitar el desarrollo y diseño de aplicaciones web [Powers00]. Apareció a la vez que el navegador Mozilla [Mozilla10], ya que está diseñado usando XUL. El lenguaje permite describir, definir el contenido de una aplicación y su distribución y presentación en la ventana de un navegador Web cualquiera. Modificando unos pocos ficheros XUL que definen el contenido de la aplicación, podemos cambiar por completo la apariencia final de la interfaz. Los ficheros de tipo CSS se utilizan para definir diferentes estilos y comportamientos deseados. Los ficheros de definición utilizados siguen las especificaciones W3C [W3C10].

La plataforma LiquidUI de Harmonía se centra en el desarrollo de herramientas usando el lenguaje UIML [Ali+02]. El lenguaje UIML (User Interface Markup Language) es un metalenguaje basado en XML para interfaces de usuario multiplataforma [Abrams+99]. UIML genera código para cualquier dispositivo sin tener que saber programar en el lenguaje requerido por el dispositivo. Gracias a UIML se pueden diseñar interfaces para cualquier dispositivo sin escribir código. En UIML, no es necesario conocer el hardware o software en el que se despliega la interfaz, basta con definir la lógica. La aplicación LiquidUI se encarga de representar la interfaz desde su definición en UIML en el código que se necesite, HTML, Java, WML, VXML, etc. La implementación de UIML se encargará de poder comunicarse con los *beans* (componente software que tiene la particularidad de ser reutilizable) para poder hacer que funcione la interfaz.

La plataforma LiquidUI hace uso de un servidor UIML que recibe peticiones http para diferentes interfaces de usuario desde navegadores tipo Wap, VoiceXML y UIML, encuentra en fichero UIML adecuado y lo envía. Cabe destacar que UIML no permite el cambio dinámico o la reconfiguración tal y como requieren los sistemas que estamos barajando.

Existen otros trabajos sobre cómo poder utilizar XML y sus variantes para el intercambio de información [Garlan+02]. La mayoría de estos trabajos están centrados en la adaptación de las interfaces de usuario como mecanismos de interoperabilidad.

Pero muchas veces, aparte de adaptar las interfaces de usuario, se necesita mover y adaptar los programas con los que se está interactuando, no siendo suficientes este tipo de técnicas basadas en XML para lograr el objetivo propuesto. Una buena interfaz de usuario debe ser adaptable y uniforme.

El Lenguaje Extensible de Formato para Aplicaciones XAML (eXtensible Application Markup Language) es un lenguaje de marcado basado en XML desarrollado por Microsoft [XAML10]. XAML forma parte de Microsoft Windows Presentation Foundation (WPF). XAML es el lenguaje que se usa para la presentación visual de las aplicaciones desarrolladas con Microsoft Expression Blend, del mismo modo que HTML es el lenguaje que se usa para la presentación visual de las páginas Web. Al ser XAML declarativo (del mismo modo que HTML), si se quiere agregar lógica en tiempo de ejecución a la aplicación se requiere añadir código. Por ejemplo, si la aplicación únicamente usa XAML, se pueden crear y animar elementos de la interfaz de usuario y configurarlos para que respondan de un modo limitado a los datos proporcionados por el usuario (mediante desencadenadores de eventos), pero la aplicación no podrá realizar cálculos ni responderlos, ni podrá crear espontáneamente nuevos elementos de UI sin añadir código. El código de una aplicación XAML se almacena en un archivo distinto del documento XAML.

Por otro lado y en la misma línea, en OpenLaszlo [OpenLaszlo10] también proponen una plataforma de desarrollo web open-source o de código abierto para el desarrollo y distribución de RIA ‘Rich Internet Applications’. Se trata de una plataforma en la que a través de un desarrollo de programas en XML (llamados LZX) podemos hacer aplicaciones ricas en gráficos preparadas para internet utilizando como plataforma de ejecución el motor de Flash y DHTML. LZX fue diseñado para utilizar las mismas convenciones y sintaxis familiares para los desarrolladores de aplicaciones web. Los programas LZX utilizan estructuras de procedimientos y declarativas a la vez que se definen formatos de nombres de tipo CSS (*Cascade Style Sheets*). En OpenLaszlo, al ser un lenguaje interpretado y ejecutado por el navegador, cabe destacar que toda la compilación se realiza en el servidor OpenLaszlo, y el archivo resultante con formato Flash (.swf) puede ser ejecutado en el navegador mediante el *plug-in* para Flash. De todas formas, LZX no utiliza el modelo de objetos internos de Flash.

Un aspecto importante relativo a las interfaces y que a veces se deja de lado es el usuario final. Una interfaz adaptable al usuario permite que el usuario pueda cambiar la manera en que interactúa con la interfaz de la aplicación. Una interfaz uniforme provee al usuario la sensación de “tal y como se ve, se maneja”. La uniformidad se aplica a la interacción dentro de una aplicación dada y las interacciones con una familia de aplicaciones. El usuario se familiariza con una aplicación y las siguientes aplicaciones que el usuario utilice deben ser similares a la ya utilizada anteriormente, logrando que el usuario no pierda tiempo en familiarizarse con cada aplicación. Cualquiera de las técnicas o metodologías que se utilice deberá adecuarse a las costumbres de uso y manejo de los usuarios finales, sin introducir cambios drásticos al menos desde un punto de vista de la usabilidad.

2.3 Configuración y personalización

Se prevé que el número de teléfonos móviles sea de 174 millones en 2011, y se espera que en esa misma fecha haya 10 veces más máquinas o dispositivos que gente en el mundo [Abi05, Abi07]. Estos dispositivos serán capaces de comunicarse unos con otros e intercambiar información entre ellos, y engloban no sólo teléfonos móviles o PDAs sino también muchos otros dispositivos que están embarcados en nuestro entorno. Quizá no los vemos, pero un creciente número de sistemas embarcados está ya en nuestras vidas. Estos sistemas se encuentran en microondas, reproductores de audio, coches, termostatos, cámaras de fotos, televisiones, semáforos, juguetes, sensores, actuadores, dispositivos electrónicos modernos y otros muchos más. Esta cantidad de dispositivos de tamaño reducido, autónomos, con razonables capacidades de computación, comunicación cableada y/o inalámbrica [Smarthome09] junto con el desarrollo de múltiples aplicaciones están permitiendo que en los entornos se detecten y ajusten/controlen las funcionalidades de los dispositivos, es decir, del propio entorno, de acuerdo con las peticiones, preferencias o configuración del usuario.

Por otro lado, nuevas tecnologías de comunicación están permitiendo más flujos de información entre dichos dispositivos. Esto posibilita la aparición de nuevos servicios basados en la localización del usuario o en el contexto, esto es, mover las aplicaciones/servicios de un usuario cómo, cuándo y dónde él quiera, acceso a información donde quiera que el usuario vaya o control remoto de los dispositivos. El resultado es que estos dispositivos no aislados y más inteligentes permiten un gran campo de aplicaciones en varios dominios, como son el industrial, el del hogar y el entorno laboral, creando entornos inteligentes.

La adaptación/configuración del entorno y personalización juegan un papel importante para los usuarios en los entornos ubicuos. En este ámbito, existen varias vertientes para personalizar o automatizar el entorno (se entiende que personalización es un término que incluye la automatización). Por ejemplo, una vertiente está basada en la deducción de patrones de comportamiento del usuario en el entorno con el fin de automatizar el entorno. Otra está basada en la definición que el usuario realiza para personalizar el comportamiento ambiental (por ejemplo, que la temperatura sea de 24 grados). En la primera, la configuración se lleva a cabo deduciendo los patrones de comportamiento de los usuarios a través del uso de sensores e información de otros dispositivos (generalmente usando mecanismos de Inteligencia Artificial) [Perez+07, Vastenburg+07, Ravi+05] y la segunda vertiente se realiza dando respuesta a las necesidades o preferencias predefinidas por el usuario [Nishigaki+05, Weis+07].

Aunque en los últimos años se ha realizado un gran esfuerzo en evaluar la información de contexto para reconocer las acciones o actividades del usuario y adaptar o reconfigurar el entorno basándose en dicho reconocimiento, los trabajos realizados caminan lentamente hacia una solución para este problema.

Algunos trabajos intentan detectar la actividad que el usuario está llevando a cabo (por ejemplo, caminar, correr, subir o bajar escaleras, limpiar los dientes, etcétera) partiendo de la información que envían diferentes sensores y acelerómetros [Headon+02, Ravi+05, Seon-Woo+02]. En [Vastenburg+07], después de realizar el análisis del comportamiento de varios usuarios, capturando información de todas las acciones llevadas a cabo durante varios días, optaron por automatizar dichas acciones. El sistema decidía teniendo en cuenta lo que ocurría en el entorno simulado de un hogar en el laboratorio con usuarios reales. Una de las conclusiones que obtuvieron fue que cuando el sistema ejecuta acciones que el usuario no espera, la confianza y la percepción del usuario se ve afectada. La toma de ciertas decisiones automáticas basadas en deducciones que pueda realizar un sistema de este tipo no siempre es la más adecuada, resultando en una merma de la confianza de los usuarios en el sistema. En este sentido, parece que en general los usuarios prefieren que las acciones que ocurran sean previsibles o predecibles en cierto modo. Por ejemplo, la acción de apagar automáticamente las luces, la televisión… debido a que el sistema no detecta actividad en la sala, no es bien percibida por los usuarios, sobre todo si el usuario se encuentra sentado en el sofá viendo tranquilamente la tele. Habrá que utilizar otros mecanismos para identificar la 'no actividad' de los lugares. En [Ravi+05] utilizan acelerómetros para detectar la actividad que está realizando la persona. Pero la identificación de acciones cotidianas como limpiarse los dientes, lavarse o abrir una puerta presenta dificultades utilizando este tipo de tecnología. En cambio en [Headon+02] mediante un suelo sensible, reconocen la posición y movimiento que el usuario está realizando. Utilizan toda esta información para interactuar con juegos. En [Seon-Woo+02] utilizan acelerómetros y técnicas de inteligencia artificial para tratar de localizar a las personas dentro de una estancia y proporcionar contenido personalizado. También en [Perez+07] tratan de reconocer la actividad del usuario utilizando secuencias de vídeo y buscando patrones de comportamiento. Este tipo de sistemas necesitan un entrenamiento y periodo de aprendizaje.

En este trabajo de tesis, las acciones serán definidas por los usuarios y, por tanto, los usuarios tendrán conocimiento de lo que tiene que ocurrir. La mayoría de estos trabajos se basan en la captura y almacenamiento centralizado del contexto y en técnicas de razonamiento con un enfoque "user-push" en la que los sistemas ejecutan las acciones sin la petición expresa de los usuarios [Cheverst+01, Feigenbaum92, Kendall+99]. En estos casos, los deseos del usuario y sus preferencias no son tenidas en cuenta. En esta tesis, se tratará de dar solución a esta necesidad, la información de contexto se usará para desencadenar acciones predefinidas por el usuario. No obstante, con un sistema en modo automático no se va a poder evitar la ejecución de acciones predefinidas que no siempre responden al deseo puntual del usuario. El modo manual y el poder actuar por encima del sistema automático deberán estar habilitados para satisfacer las preferencias puntuales del usuario. En el sistema que se propone, esta flexibilidad dependerá del tipo de reglas que se definen y del grado de automatización elegido para cada conjunto de usuarios y dispositivos.

A continuación se describe el estado del arte de las diferentes áreas de estudio, los sistemas basados en reglas, incluyendo los motores de reglas, y los editores para lograr la personalización. El motor de reglas dotará al middleware de inteligencia.

2.3.1 Sistemas basados en reglas

Para adaptar el entorno, en general se usan sistemas basados en reglas. Estos sistemas típicamente evalúan información de varios dispositivos y servicios junto con información de contexto disponible y comprueban si las condiciones de las reglas se satisfacen [Jung+07, Pietzuch+03, López-de-Ipiña01]. Esta evaluación se realiza en el motor de reglas. Tras la evaluación de las condiciones, el motor de reglas se encargará de lanzar la ejecución de las acciones correspondientes.

A continuación, se van a describir una serie de características necesarias para lograr la personalización del entorno. Algunas están más directamente relacionadas con la inteligencia del middleware framework y otras con la creación del editor TAMAmI.

2.3.2 Gestión de eventos (Eventing)

En la personalización, la gestión de eventos juega un papel muy importante. En la literatura existen numerosos ejemplos de mecanismos utilizados para la detección y distribución de los mismos a las entidades suscritas. A continuación, se describen algunos de los trabajos más relevantes.

AutoHAN es una arquitectura que permite la especificación de la interacción de los dispositivos, como son los electrodomésticos en el hogar [Blackwell+01]. AutoHAN usa GENA (*General Event Notification Architecture*) para el envío y la recepción de notificaciones de los dispositivos en la red. GENA es una arquitectura de notificación de eventos incluida en UPnP y que hace uso de HTTP sobre TCP/IP o UDP *multicast.*

También, AutoHAN posee un servicio de registro de los servicios disponibles en la red, llamado DHan. DHan usa XML para almacenar y presentar las propiedades de los servicios y a este registro se accede usando HTTP. GENA y DHan permiten que la red del hogar definida por AutoHAN sea autoconfigurable. Por ejemplo, cuando un nuevo dispositivo aparece en la red, se establece una fuente de eventos y se autoregistra en DHan (servicios descubribles). Entonces, diferentes servicios pueden buscar y acceder a estos nuevos servicios, y comunicarse vía GENA.

En el ámbito del SOA también existen protocolos como el WS-Eventing [WS-Eventing06] para que a través de servicios web se pueda suscribir con fuentes de las que se recibirán mensajes de notificación. Esta especificación define un protocolo para que un servicio web se pueda suscribir con otro servicio web para recibir notificaciones de eventos de interés.

Las arquitecturas clásicas de los motores de reglas ECA centralizaban la entrada y la gestión de todos los eventos en un único componente que a continuación evaluaba la lista de condiciones una por una, incluso si la variable involucrada no estaba en la

condición, intentando buscar correspondencias entre el evento que había llegado y las condiciones definidas.

En [Pietzuch+03] proponen un *framework* en el que también utilizan un mecanismo publicador/subscriptor para la asociación de variables lógicas con los eventos producidos por los dispositivos externos. Concretamente, se centran en el análisis de eventos compuestos. Denominan eventos compuestos al conjunto de patrones que desean identificar en el entorno. De manera análoga con este trabajo de tesis, los eventos compuestos se asemejan a las reglas con sus condiciones compuestas. Lo que denominas como eventos primitivos son la base de comunicación de todo sistema que utiliza un mecanismo publicador/subscriptor. Nuestro trabajo de tesis también se basa en los mismos eventos primitivos, pero el manejo y la gestión posterior de dichos mensajes se realizará de manera diferente a los autómatas que utilizan, tal y como se explica en el capítulo 4.

En [Beer+06], han diseñado un guía turístico utilizable en museos o para visitar ciudades que, basándose en la localización de los usuarios, propone diferentes rutas y ayuda en la toma de decisión a los usuarios. Proponen la utilización de eventos tipo 'push', es decir, el usuario se suscribe a los eventos que quiere ser notificado, y a posteriori en función de las reglas definidas ya se verá qué hacer con los mismos. Más concretamente, la asociación con el evento se realiza a nivel de la condición. El gestor de eventos ('Event-Handler') es el que conoce y controla la asociación entre el evento que llega y que condición/condiciones hay que evaluar. Como se verá más adelante, nuestra aproximación empareja eventos directamente con las variables involucradas en las propias condiciones, siendo las propias variables las que desencadenan la ejecución de la evaluación de las condiciones y las reglas asociadas. Aunque ambos sistemas utilicen como base un sistema de eventos tipo 'push', la posterior gestión de dichos eventos se realiza de manera diferente.

En trabajos tan diferentes como [Zhang+05, McBurney+07, Dongliang+08, Chen+05], se da como válida la utilización de mecanismos conjuntos de *eventing* y paradigma ECA. Sirvan todos ellos como muestra de la utilización de los mismos principios básicos de eventos para dar solución a escenarios y ámbitos de aplicación diferentes, desde el control de redes de sensores inalámbricos hasta el control de invernaderos, pasando por el control y optimización de la red eléctrica en Shanghái.

2.3.3 Sistemas basados en eventos

En esta sección, se describen diversos trabajos que muestran la utilización, validez e importancia de los sistemas basados en eventos.

El uso de eventos puede ser muy importante o crítico en algunas situaciones, para mejorar el rendimiento de un sistema y para definir el momento de actuación del mismo. En [Dayu+05], describen un esquema basado en eventos para solucionar el problema del elevado número de lecturas que un lector RFID tiene que hacer en un almacén al gestionar el número de objetos existentes. La solución propuesta consiste en leer las etiquetas RFID de únicamente los objetos que cambian de posición (salen o

entran del almacén). La actividad de los lectores está desencadenada por el movimiento de un objeto, es decir, por un evento. Así pues, el tráfico de lecturas y el tiempo de procesamiento de los lectores RFID (con ello, la durabilidad de la batería) se optimiza.

En [Ridong+09] utilizan una arquitectura basada en eventos y reglas para crear un sistema de diálogo para un robot. En función del evento detectado, el sistema procesa las reglas prefijadas en el sistema y el robot responde a lo solicitado. Diferentes componentes forman esta arquitectura y la comunicación entre los mismos está basada en eventos. Esta arquitectura reduce el tiempo de procesamiento global, puesto que no existen bucles cíclicos donde se analiza línea por línea si se satisface una condición o no, directamente se evalúan únicamente lo relativo al evento ocurrido.

En nuestro caso, el motor de reglas propuesto también hace uso de los eventos para mejorar su rendimiento y respuesta, como se verá a continuación.

2.3.4 Emparejamiento y ejecución de las reglas: motor de reglas

Tradicionalmente, cuando se producían eventos, los motores de reglas evaluaban cíclicamente un conjunto de condiciones de las reglas concatenadas IF-THEN [Pietzuch+03, Beer+06, Guerrero+08, López-de-Ipiña01, Zhang+04]. Esta evaluación se llevaba a cabo incluso si el evento recibido no estaba asociado a una condición, es decir, incluso si el evento no formaba parte de ninguna regla.

El motor de reglas descrito en [Beer+06] consiste en un manejador de eventos, un evaluador de condiciones y un gestor de acciones. El manejador de eventos se encarga de notificar de los eventos a las reglas que se hayan suscrito. Esto se realiza a través de un algoritmo de emparejamiento que compara los mensajes de eventos de entrada con un perfil de eventos definido para cada regla. Esto implica comprobar todos los perfiles de eventos en un orden dado, lo que puede afectar al rendimiento.

Los autores de [Pietzuch+03] proponen un *framework* para manejar un número elevado de eventos. En este trabajo, el *framework* propuesto se puede considerar que es un motor de reglas, y lo que ellos llaman eventos compuestos (*Composite Events – CE*) serían las reglas. Esto es, en lugar de abordar el problema de emparejamiento de los eventos y las reglas, analizan el emparejamiento de eventos y CEs. Este emparejamiento se lleva a cabo a través de unos detectores, que son simplemente autómatas de estados finitos con soporte temporal. De nuevo, la naturaleza de este emparejamiento refleja la secuencia lineal de la búsqueda.

En la misma línea otro trabajo [Parra+09] propone la utilización de reglas ECA por los diferentes dispositivos que utilizan la arquitectura *Device Profile for Web Services* (DPWS) para el intercambio de información entre iguales *Peer-To-Peer* (P2P). En cuanto a la personalización y facilidad de cambio de comportamiento del sistema, no especifica ningún mecanismo ni la utilización de herramientas dirigidas al usuario para la creación y ejecución de dichas reglas ECA. Por lo que la adición del comportamiento deseado se lleva a cabo en el mismo código de ejecución de cada

dispositivo, siendo muy laboriosa la modificación posterior del mismo. El dinamismo entre dispositivos está bien resuelto por su parte en cuanto a la heterogeneidad e interoperabilidad. No en cambio la adaptación y configuración de las reglas que sigue el sistema. La posterior personalización modificando y adecuando a los diferentes escenarios de aplicación puede llegar a ser una tarea complicada, sin la utilización de un editor de reglas y un motor o componente que aglutine el comportamiento de cada dispositivo que actualmente funcionan independientemente.

En [Guerrero+08] se describe la evaluación de un motor de reglas. En este motor de reglas, la asociación entre eventos y reglas se lleva a cabo definiendo un *path* o camino. En ese trabajo, un camino se define como una secuencia de servicios ECA que un evento sigue para ejecutar o no una regla. Los servicios ECA pueden ser: composición de eventos, evaluación de condiciones y ejecución de acciones. Por ejemplo, un camino implica la evaluación de todas las condiciones de las reglas y sus reglas si las condiciones son ciertas, lo que de nuevo muestra sus desventajas cuando existe un gran número de reglas.

La misma idea se aplica al servicio de emparejamiento de reglas ECA basado en CORBA descrito en [López-de-Ipiña01]. En dicho trabajo, el motor de reglas Jess (*Java Expert System Shell)* realiza el razonamiento basado en reglas [Friedman-Hill09]. Otros autores también han hecho uso de Jess y de su gestor de eventos [Zhang+04].

Jess es un motor de inferencia escrito en Java y embebido en un servidor ECA que lleva a cabo el razonamiento basado en las reglas. Jess usa una versión mejorada del algoritmo RETE para procesar las reglas. El algoritmo RETE es un mecanismo muy eficiente para resolver problemas de emparejamiento múltiple[2] o igualación de múltiples patrones y evitar la evaluación de cada una de las reglas [Forgy82]. Al interpretar las reglas, el algoritmo RETE construye una red (llamada red Rete) en forma de árbol que contiene todos los patrones de las condiciones (una sola vez), evitando ciclos internos de igualación de los eventos (o hechos) con todas las reglas. Usando dicho árbol y guardando información temporal de las evaluaciones, el algoritmo RETE busca exclusivamente los cambios en las correspondencias de cada ciclo, mejorando el coste computacional. Como consecuencia, la respuesta del sistema global será mejor. Otra de las ventajas que presenta este algoritmo es que elimina la duplicación de reglas.

Las principales desventajas son el gran tamaño de la información almacenada, el alto coste de operaciones de borrado y modificación, la prioridad dada a las reglas de creciente creación y que no soporta restricciones de tiempo [Walzer+08]. Dado que las

[2] En inglés '*many-to-many matching'*

aplicaciones ubicuas involucran frecuentemente eventos temporales, el algoritmo RETE no parece adecuado. Por otro lado, las aplicaciones ubicuas pueden contener un gran número de reglas aunque, en un principio, podría ser un número menor para lo que el algoritmo RETE fue diseñado.

Otros ejemplos similares a los descritos se encuentran en el dominio de los sistemas expertos. Lenguajes declarativos como Prolog [Bratko00] o CLIPS (*C Language Integrated Production System)* [NASA99] se han usado de manera satisfactoria en el contexto de programación lógica y sistemas expertos basados en reglas. Ambos lenguajes, Prolog y CLIPS, permiten representar reglas y tienen un motor de reglas implementado sobre un intérprete del lenguaje. Dado un conjunto de hechos (proposiciones verdaderas de las entidades) y reglas aplicadas a ellos, un motor de inferencia o motor de reglas decide qué regla debe ejecutar o lanzar.

A diferencia de los trabajos anteriores, el motor de reglas propuesto en este trabajo de tesis, pretende dar solución a los problemas detectados sobre la evaluación de las condiciones de las reglas ante la presencia de eventos y las desventajas de la ejecución cíclica del grupo de reglas. En lo que se refiere al problema de evaluar solamente las reglas cuyas condiciones se cumplen, como se verá, la solución aportada consiste en usar un mecanismo particular basado en eventos (concretamente en un entrelazado de eventos). Los cambios en el contexto generarán eventos, y únicamente las condiciones relacionadas al producirse dichos eventos serán evaluadas. Esto se conseguirá con el uso de variables lógicas o virtuales asociadas a las variables reales. El mismo mecanismo de eventos se usará entre las condiciones y las acciones de las reglas. Una vez se satisface una condición, se genera un evento que será consumida por la acción o las acciones de las regla correspondientes.

Este mecanismo se ha usado satisfactoriamente en otros dominios, como en los de procesos de producción [Schiefer+07] y en trabajos similares [Herbert+08, Nishigaki+05].

Como contribución importante de esta tesis, se propondrá la utilización de un mecanismo de activación de reglas basado en la ejecución particular de cada regla en función de si alguna parte de dicha regla (condición) se ha modificado y necesita ser reevaluada. El modelo Publisher-Subscriber juega un papel importante para lograrlo de una manera.

2.3.5 Detección de conflictos

Un tema importante a considerar es la posibilidad de definir prioridades en las reglas para el caso de que exista un conflicto (por ejemplo, dos reglas desencadenen acciones contradictorias).

En los años 80, el algoritmo RETE propuso un mecanismo de resolución de conflictos [Forgy82]. Para ello, este algoritmo representa la implementación lógica de la funcionalidad responsable para el emparejamiento de eventos y reglas. Durante este emparejamiento, el motor encuentra todas las parejas para los eventos y determina un

orden en el que las reglas o acciones se pueden ejecutar. Esta última tarea se llama resolución de conflictos y, en realidad, no está definida como parte del algoritmo RETE aunque se usa conjuntamente. Por ejemplo, CLIPS tiene implementado también un mecanismo para la resolución de conflictos, donde los usuarios pueden definir las prioridades de las reglas y manejará los conflictos de acuerdo al orden establecido.

Una revisión de la literatura sobre esta cuestión muestra que las prioridades de las reglas es la estrategia más usada [Nishigaki+05, Shankar+05, Shankar+06, Zhang+04, Li+09, Lee+07], aunque muchos autores no abordan este problema [Beer+06, Guerrero+08]. Existen también otros mecanismos para abordar el problema de la detección de conflictos. En [Lee+07] además de usar prioridades, por ejemplo, proponen la utilización de *locks* o semáforos para acceder a los recursos o variables, de tal forma que en el momento que se quiera modificar una variable, si previamente está bloqueada el gestor de los *locks* decidirá qué hacer . En caso de conflicto, el gestor de los *locks* decidirá qué petición debe esperar y cuál ser ejecutada dependiendo de las prioridades de los servicios (no de las reglas). Por ejemplo, los servicios relacionados con la seguridad tendrán mayor prioridad que los de entretenimiento. Así pues, cuando ocurre un conflicto y dos servicios quieren acceder al mismo dispositivo o cambiar el valor de una variable, el gestor comprueba cuál de los dos tiene mayor prioridad. Todas las peticiones de invocación de servicios son controladas por el gestor para detectar futuros conflictos.

El trabajo anterior trata de dar solución a los conflictos que puedan surgir en un entorno ubicuo donde diversos dispositivos están conectados y varios usuarios acceden a los mismos. Aunque en un dominio diferente y, por tanto, sin ser necesario el uso de *locks*, este trabajo de tesis abordará la detección de conflictos de una forma muy similar (apartado 5.4.2.1 Resolución de conflictos).

El trabajo en [Jung+07] sobre la resolución de conflictos (o como ellos llaman inconsistencias potenciales entre las reglas) es muy similar al aquí propuesto. Existen dos casos en los que se lleva a cabo la resolución de conflictos. Por un lado, se detectan los posibles conflictos al cargar una regla en el motor. En caso afirmativo, no dejan añadir la regla al grupo existente. En el segundo caso, se detectan los conflictos en tiempo de ejecución. Estos conflictos en tiempo de ejecución se resuelven utilizando otros mecanismos prescritos de manera predefinida, no queda claro cuál o cuáles son las estrategias utilizadas. Está claro que, al igual que en este trabajo, habrá que abordar la resolución de conflictos en ambos casos.

Por otro lado, la forma de abordar la resolución de conflictos, también es peculiar. Por un lado, defienden que no se puedan crear ciertas reglas si se detectan coincidencias a la hora de la creación de las mismas y, por otro, si en tiempo de ejecución se activan varias reglas simultáneamente, dejan en manos del usuario la priorización de las mismas. En nuestro caso, la detección de conflictos se plantea de manera diferente. Por un lado, se pretende priorizar la ejecución de reglas en función del perfil del usuario. Es decir, los miembros de una casa van a tener asignada una prioridad diferente, de tal forma que cuando entren en conflicto reglas de usuarios diferentes, el

sistema haga caso a aquella regla que tiene mayor prioridad. En tiempo de ejecución, todos los eventos van a ser evaluados acorde a su prioridad, y si la acción resultante entra en conflicto con alguna otra regla existente con prioridad mayor, se evaluará de tal forma que se satisfaga el de mayor prioridad (véase apartado 5.4.2.3 Resolución de conflictos). En caso de coincidir la prioridad entre dos reglas, la ejecución de las acciones se realiza en el orden en el que han ocurrido, es decir, el resultado final es como si no hubiera existido ningún conflicto, prevaleciendo por así decirlo la última regla satisfecha.

Existe otra aproximación para abordar la resolución de conflictos y, aunque parezca contradictoria, dicha teoría defiende la no utilización de motores de reglas para resolución de conflictos en tiempo de ejecución [Hicks07]. En este caso, defienden que son los desarrolladores de aplicaciones los que conocen el tipo de conflictos y condiciones que tiene que resolver el motor de reglas, y que ese conocimiento puede ser plasmado en el momento de la codificación, liberando al motor de reglas de ciertas tareas automatizadas.

2.3.6 Adaptación o configuración

Tal y como se ha mencionado anteriormente en los trabajos [Zhang+05, McBurney+07, Dongliang+08, Chen+05] gracias al *framework* el motor de reglas adaptará sus recursos basándose en los existentes en la red. Cabe destacar que en [Becker+04a] han concebido una infraestructura genérica para la creación de aplicaciones configurables y adaptables para prácticamente cualquier tipo de escenario. En concreto, la interoperabilidad entre dispositivos en [Becker+04a] se basa en un trabajo anterior de los propios autores denominados BASE [Becker+03]. En este trabajo de tesis, los dispositivos se comunicarán y apoyarán en los protocolos que se han especificado en el capítulo 3, evitando protocolos propietarios y no estandarizados. Por otro lado, la creación de reglas para la configuración del entorno queda en manos de los programadores, no pudiendo los usuarios crear dichas reglas para poder personalizar con sus preferencias. La adaptación se lleva a cabo solo a nivel del sistema una vez que las reglas ya han sido cargadas en el sistema. En esta tesis, abordaremos la adaptación del sistema a las necesidades concretas del usuario, y se propondrá el uso de un editor configurador para que el usuario pueda generar las reglas necesarias para lograrlo. En el Capítulo 6 se explica la solución propuesta.

2.3.7 Identificación y agrupación de tipos de servicios similares

En primer lugar, la identificación y clasificación de diversos servicios homogéneos juega un papel importante en la creación de reglas. A la hora de definir las reglas para personalizar el comportamiento de un entorno, parece fundamental que la información de la parte de la condición de dichas reglas pueda hacer referencia a un tipo o a una categoría del servicio. Una categoría de servicio servirá para agrupar los espacios de servicios que puedan existir en un entorno de acuerdo a su funcionalidad, esto es, agrupar los dispositivos por las funcionalidades que ofrecen. Así pues, las reglas que involucren la invocación de un tipo de servicio ofrecido por más de un dispositivo se

podrán definir más fácilmente. Por ejemplo, la regla 'que se cierren todas las persianas de casa cuando hay tormenta' hace referencia a todos los servicios de tipo 'cerrar persianas'. Claramente, se ve la necesidad de identificar o agrupar los dispositivos por las funcionalidades que ofrecen.

Para identificar y agrupar los dispositivos y servicios en función de la funcionalidad ofrecida, algunos protocolos de interoperabilidad definen sus propias categorías de dispositivos y servicios. Este es el caso de UPnP [UPnP-desc10]. UPnP tiene definidas ciertas categorías (por ejemplo, audio/vídeo, luz, impresora, *gateway* o pasarela de comunicación, *router* WLAN, automatización del hogar) y describe los tipos de dispositivos y servicios de una forma estandarizada dentro de cada categoría. Por ejemplo, dentro de la categoría de 'automatización del hogar' existen una subcategoría que es 'SolarProtectionBlind:'. Esta a su vez consta de un dispositivo 'SolarProtectionBlind' que ofrece un único servicio 'TwoWayMotionMotor'. UPnP define el dispositivo y el servicio con sus correspondientes variables de estado, *eventing* y acciones asociadas. En este sentido, se puede ver que se está avanzando en la estandarización de la descripción de los servicios y dispositivos, aunque creemos que todavía no hay nada maduro que integre la diversidad existente. Por ejemplo, la descripción de los diversos sensores y actuadores no está contemplada. Por otra parte, un mismo dispositivo podría ofrecer servicios pertenecientes a varias categorías, no considerado hasta la fecha. En este sentido, se han generado nuevas categorías de servicios que se van a utilizar en este trabajo de tesis (véase Anexo A dispositivos virtuales UPnP para más detalle).

2.3.8 Editores para la personalización

La personalización de un entorno puede llegar a ser una tarea complicada y a veces difícil de realizar. La inclusión de una interfaz para poder hacerlo puede ser vital tanto para un usuario con perfil administrador/instalador como para un usuario final. En este ámbito, es necesaria la utilización de editores que faciliten la generación de reglas ECA, ayudando al usuario final, incluso al personal de mantenimiento de un entorno físico (almacén, oficina, fábrica, hospital, etc.) que normalmente carecen de conocimientos en programación.

Dentro los trabajos que abordan la creación de reglas, se analizarán primero los sistemas o trabajos no basados en reglas ECA y, a continuación, los que hacen uso de las mismas.

2.3.9 Sistemas no basados en reglas ECA

Existen varios trabajos que se centran en el diseño personalizado o modelado de aplicaciones ubicuas sin hacer uso explicito del paradigma ECA. Algunos de ellos se han centrado también en la interacción del usuario en el diseño de dichas aplicaciones para que sea sencilla y fácil [Blackwell+01, Weis+07, Li+04].

Uno de los primeros trabajos que aparece en la literatura fue [Blackwell+01], donde se describe AutoHAN. AutoHAN es una arquitectura que permite la especificación de la

interacción de los dispositivos, como son los electrodomésticos en el hogar. En este trabajo se detallan las demandas cognitivas del usuario doméstico tomadas en cuenta para el diseño de la interfaz gráfica. Por ejemplo, se destaca sobre la importancia de manejar bloques físicos (manipulación directa y real de objetos), no simplemente representaciones en la pantalla del ordenador. También, se describe el reto que es desarrollar un interfaz de un dispositivo que incluya elementos que no se pueden ver ni tocar, la notación abstracta. Así pues, en este trabajo se definen dos paradigmas (Ontológico y Lingüístico) de interacción dónde los Media Cubes (cubos físicos) pueden ser asociados físicamente con dispositivos concretos. Estos cubos físicos pueden ser considerados como el lenguaje de programación para los usuarios domésticos y funcionan como un botón de control remoto. Los cubos están formados por una variedad de sensores y transductores, un microprocesador, pilas y un sensor de movimiento. Cada cubo contiene un botón y una luz LED que se usan para confirmar el estado y un transductor eléctrico que se usa para notificar al usuario de los cambios locales.

Además, estos cubos están equipados con una red de emisores y transmisores infrarrojos para comunicarse con el resto de los elementos de la red y con bobinas de inducción en cuatro de las caras del cubo que actúan como antenas para detectar la proximidad de otros cubos y para establecer una relación entre un dispositivo y un cubo. Al situar la cara de un cubo sobre un electrodoméstico, se asocia el cubo con alguna función del electrodoméstico. Así el cubo se convierte en una interfaz para ejecutar una acción o función del electrodoméstico. Por ejemplo, un cubo puede representar la función ‘reproducir/parar’ de un reproductor de DVDs, y al presionar el único botón de ese cubo reproduce o para el vídeo. En el primer paradigma, los cubos identifican conceptos que denotan una correspondencia visual entre las operaciones de control y las funciones de un dispositivo (Paradigma Ontológico), mientras que en el segundo los cubos representan palabras (Paradigma Lingüístico). Aunque es un enfoque interesante, el resultado de este tipo de interacción limita su aplicación a un determinado número de eventos y acciones y el sistema es altamente dependiente del hardware.

Otro trabajo similar se centró en un lenguaje visual de alto nivel, llamado VisualRDK, para el desarrollo de aplicaciones ubicuas por desarrolladores [Weis+07]. Dicho lenguaje contiene los controladores de ciertos eventos definidos a priori y proporciona una serie de comandos para definir la lógica de una aplicación ubicua. El resultado es una herramienta de difícil manejo para usuarios no desarrolladores. A diferencia del trabajo anterior, el grupo de investigadores en [Li+04] se centran en la interacción del usuario final al diseñar una aplicación ‘*location-enhanced’* (por ejemplo, encuentra las impresoras más cercanas). La herramienta propuesta usa un mayor nivel de abstracción, lo que permite a un diseñador pensar visualmente en referencia a la localización de las personas, lugares, mapas, etc. Sin embargo, su uso está limitado al desarrollo de aplicaciones de tipo búsqueda de información (‘browsing information’) dejando fuera las aplicaciones del tipo automatización o ‘follow-me’.

También muy centrado en el desarrollo de aplicaciones ubicuas por desarrolladores es el trabajo propuesto en [Becker+04b, Weis+06]. A través de Nexel, un lenguaje de programación gráfico, los usuarios pueden crear aplicaciones ubicuas. Por ejemplo, utilizando su plataforma 'PCOM', es posible la coordinación de los distintos servicios existentes. Precisamente, es posible definir que se controle el volumen de un reproductor a través del teléfono móvil. Este editor permite la creación de ciertas aplicaciones, pero no aborda aspectos como la localización o el tiempo. Tampoco se plantea qué ocurriría en caso de conflicto cuando varios usuarios están intentando acceder al mismo dispositivo.

Otro enfoque diferente a los anteriores autores propone un sistema basado en la tecnología de agentes para la personalización de entornos [Lee05]. En este último trabajo, el autor propone la generación de una serie de preferencias iníciales utilizando una herramienta gráfica y, a continuación, el sistema basado en dichas preferencias y deduciendo del árbol de decisiones correspondiente, proporcionará al usuario la información requerida. En este tipo de enfoque, aparte de ser una aplicación cerrada y específica para una serie de escenarios, nos encontramos con que la personalización llevada a cabo está más relacionada con aspectos de visualización y modificación de lo que se le muestra al usuario que con la personalización y reconfiguración real del escenario.

2.3.10 Sistemas basados en reglas ECA

Existen en la literatura varias propuestas basadas en el uso de reglas ECA para dar solución a la personalización de un entorno. Estos trabajos se centran en el desarrollo de editores gráficos para la creación de reglas ECA [Beckmann+03, Zhang+04, Beer+07, Olmedo-Aguirre+08]. El sistema *SiteView* fue uno de los primeros trabajos basados en reglas que se centraron también en crear una interfaz intuitiva y tangible para que los usuarios finales pudieran definir acciones de automatización [Becmann+03]. A través de unos objetos de control denominados 'interactors' que representan ciertas condiciones (es decir, hora, día de la semana) y acciones (por ejemplo, encender la luz), los usuarios podían colocarlos en una representación real a pequeña escala de un entorno y generar las reglas. La dependencia del uso de este hardware para diseñar las reglas, las limitadas condiciones y acciones que pueden ser representadas, junto con que solo se pueden componer hasta tres condiciones y asociar una única acción muestran los puntos débiles del sistema.

En [Zhang+04], se describe un sistema basado en reglas ECA y centrado en el diseño de aplicaciones para usuarios sin experiencia. En este trabajo, los autores proponen dos interfaces de usuario para definir reglas simples. Mientras que uno de las interfaces contiene únicamente texto y toda la información está agrupada según su rol, la otra interfaz es visual. Esta última adopta el enfoque de arrastrar y soltar ciertos elementos con forma de pieza de puzle para que los usuarios puedan crear sus propias reglas. Afirman que cualquiera de las interfaces es todavía difícil de usar por

un usuario no experto, aunque la interfaz visual redujo considerablemente el tiempo requerido para crear reglas.

En [Beer+07], se describe un sistema llamado CAIPS *(Context-Aware Information Push Service)* que ofrece información al usuario de una determinada situación contextual. CAIPS se basa en la creación pasiva de reglas donde los eventos y las acciones son estáticos, conocidos de antemano. Por lo tanto, este sistema no aborda el problema del dinamismo en la computación ubicua. Por otro lado, el sistema Mobi-D *(Model-Based Interface Designer)* se centra en el diseño y la creación de interfaces. Dicho sistema no aborda el problema de personalizar entornos ubicuos, pero algunos de los aspectos han sido un aporte valioso en esta línea de trabajo [Puerta+97].

Aunque el planteamiento y punto de partida del trabajo [Jung+07] es muy similar al que se propone en esta tesis, ellos no abordan la creación de reglas desde el punto de vista del usuario. La creación de reglas ha de realizarse en el entorno de programación, y para cada modificación es necesario recompilar de la aplicación del motor de reglas. Es necesario un editor de reglas enfocado a los usuarios finales para poder crear reglas de manera fácil y sencilla, además de un mecanismo de carga y descarga de reglas en caliente para el motor de reglas. Tal y como se ha mencionado en otros trabajos, la creación de reglas por los usuarios finales de forma fácil e intuitiva mediante un configurador gráfico, y la activación y ejecución de las mismas por el motor de reglas en función de las necesidades del momento son otras de las características que se echan de menos en este trabajo.

Existen también otra serie de trabajos que están totalmente dirigidos hacia usuarios expertos en programación y notación UML para la creación de reglas ECA [Olmedo-Aguirre+08].

Como conclusión, se podría decir que todas estas herramientas o bien no soportan todo tipo de aplicaciones ubicuas o bien fallan en el limitado poder expresivo o están centradas únicamente hacia un usuario con experiencia. A diferencia de estos trabajos, el objetivo de esta tesis es proponer un editor para que de una manera sencilla y fácil permita a un grupo de usuarios finales personalizar el comportamiento de su entorno (hogar, oficina almacén, fábrica, hospital, etc.). Además, se tratará de cubrir el amplio rango de aplicaciones ubicuas o servicios personalizados.

Capítulo 3: Interoperabilidad

3.1 Introducción

La computación ubicua y las redes asociadas han sido estudiadas desde muchas perspectivas y, en particular, por diferentes asociaciones de científicos e ingenieros [UbiComp10] [PerCom10]. Para centrar nuestra investigación y desarrollo dentro de la diversidad de retos que presenta la computación ubicua, consideramos que su objetivo debe ser el establecimiento de los canales y medios necesarios para garantizar la interacción ubicua entre usuarios, redes y dispositivos heterogéneos. La disponibilidad de los recursos de comunicación es un aspecto importante y crítico para su modo de funcionamiento. Especialmente para esos casos en los que los dispositivos móviles están en un entorno de red cambiante, la capacidad de adaptación del entorno a los recursos existentes se ha convertido en un elemento clave para proveer de una interacción ubicua y transparente para el usuario.

La utilización masiva de tecnologías baratas y accesibles de comunicación, las nuevas capacidades de computación en las propias infraestructuras de red y los dispositivos heterogéneos conectados a dichas redes han traído consigo una serie de nuevos problemas y retos a superar para los usuarios finales. En primer lugar, el reto de cómo localizar, conectar e interactuar con una red de dispositivos, en particular, de forma natural y transparente (1). Por otra parte, la proliferación de aplicaciones móviles y ubicuas ha creado a su vez, la necesidad de crear aplicaciones distribuidas capaces de adaptarse dinámicamente a los recursos existentes en cada momento (2) y, en tercer lugar, que dichas aplicaciones se adapten a las necesidades o preferencias de los usuarios (3).

En este capítulo de tesis, se describe un *middleware framework* basado en la extracción de la descripción funcional de los servicios disponibles en redes heterogéneas, que posteriormente es expuesto con una sintaxis unificada, y a su vez es ofrecido a las aplicaciones, utilizando formatos estándares de protocolos de interoperabilidad. Mediante el *middleware* propuesto se pretende dar solución a la problemática planteada en (1). La interoperabilidad a nivel de dispositivo es imprescindible para poder realizar aplicaciones que cumplen los requisitos (2) y (3).

En el capítulo 2 ‘estado del arte’ en el apartado de interoperabilidad, se ha presentado el estado del arte relacionado con el reto (1). En el apartado 3.3 se describe el *middleware* que será la base de este trabajo de tesis. Primero, se abordará la parte relativa al *middleware* en lo referente a la interoperabilidad y heterogeneidad entre dispositivos. A continuación, en el apartado 3.3, se estudiarán las características que el *middleware* deberá tener para facilitar la personalización de entornos y configuración de los dispositivos involucrados. Finalmente, en el apartado 0 se describirán las ventajas del *middleware* propuesto.

3.2 Planteamiento del problema

Los trabajos relacionados con el tema de este capítulo abarcan distintos ámbitos, tales como modelos de intercambio de información entre capas, abstracción del contexto en redes heterogéneas, mecanismos para la adaptación de aplicaciones y redes, gestión de la movilidad en redes heterogéneas, etc. Dada su amplia extensión, esta sección, describe una selección no exhaustiva de trabajos relacionados con las áreas anteriormente expuestas que se han considerado más relevantes.

En este sentido, el descubrimiento de servicios en entornos ubicuos se diferencia del descubrimiento de servicios en entornos distribuidos tradicionales. Un servicio es un elemento funcional proporcionado por una aplicación o dispositivo para que otros dispositivos o aplicaciones en la red la puedan invocar y utilizar. Los servicios y aplicaciones orientados a servicios van a ser la base en la que se va a sustentar el envío de información entre aplicaciones, es por ello que, a lo largo de esta tesis, el término aplicación va a denotar aplicación orientada a servicio. Tradicionalmente la llamada de funciones en conjunto con el uso de servicios web se lleva a cabo realizando llamadas concretas a direcciones completas y bien conocidas. Es decir, la dirección de red IP, la ubicación y dirección interna del contenido al que se desea acceder debían ser conocidas. Si por lo que fuera alguno de los parámetros anteriores no es conocido o accesible en un momento concreto, el sistema entraría en un estado de fallo. Para poder superar estos estados de fallos, se han mejorado los mecanismos de descubrimiento y búsqueda de los servicios web dinámicamente por los sistemas sin necesidad de intervención humana. El descubrimiento de servicios se compone de la capacidad de describir, diseminar, seleccionar y utilizar un servicio [Friday+04]. El descubrimiento de servicios es importante en la computación ubicua ya que los servicios pueden no ser siempre discernibles para el usuario de forma convencional, debido tanto a características físicas como a la localización virtual de los mismos. En el fondo, un sistema ubicuo deberá proporcionar los servicios de la manera más discreta posible [Weiser91].

Es la funcionalidad para ofrecer servicios en un entorno la que se pretende explotar a lo largo de esta tesis, y en este capítulo en particular. La coexistencia de varios protocolos de descubrimiento de servicios (SDP) con su propio lenguaje de descripción de los mismos convierte a los dispositivos en huérfanos si un dispositivo no cuenta con la implementación o desarrollo para cada protocolo, ya que si solo implementa uno de ellos no sería accesible por el resto. Es decir, no sería universalmente accesible. En este aspecto, este trabajo pretende proveer un mecanismo uniforme para que los dispositivos puedan ser utilizados desde cualquier aplicación que haga uso de cualquier protocolo de interoperabilidad existente. De esta manera, no importará el sistema operativo, la plataforma de ejecución o el protocolo de descubrimiento que un dispositivo de cierto fabricante ofrezca mediante los mapeos pertinentes realizados por el *middleware*, las aplicaciones podrán reconocer los servicios de dispositivos independientemente también del protocolo de descubrimiento que implemente.

Este mecanismo de virtualización de los dispositivos ofrece flexibilidad y libertad a la hora de programar y generar nuevas aplicaciones, siendo menos intrusiva y agresiva la utilización de nuevos dispositivos en entornos de desarrollo conocidos. La flexibilidad que ofrece es otro factor importante, ya que un dispositivo puede llegar a ser utilizado o invocado utilizando diferentes protocolos simultáneamente. Así pues, los usuarios y programadores de aplicaciones no necesitarán implementar su aplicación usando diferentes protocolos y podrán acceder a todos los recursos del entorno (servicios) de una manera transparente.

Aunque actualmente los fabricantes de dispositivos están asociándose y tratando de estandarizar los protocolos de acceso a sus familias de dispositivos, la realidad es que todavía no existe un estándar para el amplio rango de dispositivos. Por ejemplo, existen estándares diferentes para dispositivos médicos en el hogar 'Continua Alliance' [Continua10], para dispositivos electromecánicos el estándar está definido por la asociación IEC [IEC10], o la fundación OPC [OPC10]. Estas dos últimas organizaciones promueven la estandarización de comunicaciones en el campo del control y supervisión de procesos. Desde el punto de vista de las aplicaciones, dichos grupos de dispositivos siguen viéndose como islas de dispositivos, con las mismas necesidades de generalización y unificación que hace unos años existían en una misma isla de dispositivos. De alguna manera, lo que hace unos años eran los dispositivos de cada fabricante con sus particularidades, hoy en día se han convertido en familias de dispositivos (por ejemplo, automoción, médicos, domóticos) con sus particularidades, por lo que la necesidad de interoperabilidad a la hora de utilizarlos y gestionarlos conjuntamente dentro del hogar sigue vigente.

3.3 Descripción del *MIDDLEWARE*

El *middleware* propuesto [Uribarren+06b] ha sido diseñado fundamentándose en los siguientes principios: primero, ser capaz de obtener los servicios disponibles en el entorno heterogéneo de dispositivos; segundo, extraer la información de cada servicio en términos de descripción de sintaxis (esto es, métodos, propiedades y eventos) y meta información (por ejemplo, tipo de servicio, nombre, identificador, proveedor, etc.); y tercero, ofrecer los servicios disponibles utilizando diferentes protocolos de descubrimiento de servicio (por ejemplo, UPnP Servicios Web, etc.). La parte clave del *middleware* es el mecanismo que permite el reconocimiento y uso de un servicio ofrecido por un dispositivo activado con un protocolo de descubrimiento distinto al empleado. Dicho mecanismo se describirá a continuación. Asimismo, las posibilidades de utilización de los dispositivos (servicios) por varias aplicaciones aumentan, facilitando la creación de entornos ubicuos atendiendo a la demanda del entorno y usuario.

La figura 3.1 ofrece una visión general de la arquitectura del *middleware* y su integración entre los dispositivos y las aplicaciones. Gracias al *middleware* propuesto, las aplicaciones orientadas a servicios pueden descubrir y hacer uso de servicios

ofrecidos por los dispositivos heterogéneos presentes en la red, sin importar la tecnología subyacente (esto es, tipo de red, protocolo, plataforma).

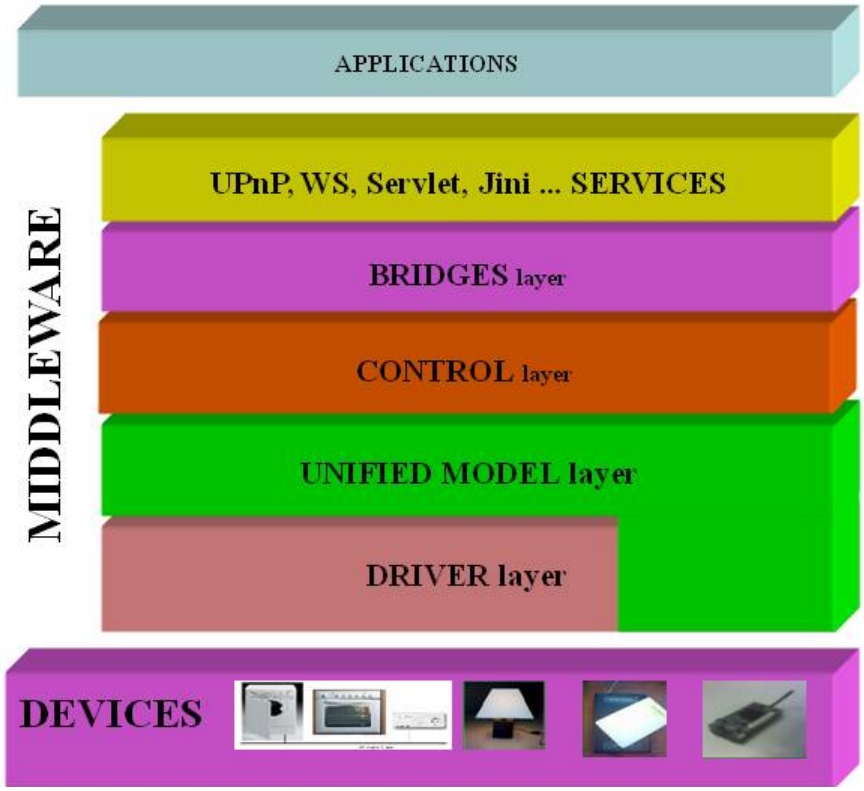


Fig. 3.1 Vista del *middleware* entre dispositivos y aplicaciones

En la figura 3.1, empezando por el más bajo nivel, la capa de Drivers se encarga de dar acceso al dispositivo, extrayendo la descripción funcional de los servicios de los dispositivos disponibles. En la capa 'Unified Model' o modelo unificado se generará una completa descripción de cada uno de los servicios mediante una sintaxis unificada.

Los servicios pueden caracterizarse mediante los siguientes elementos básicos, tal y como se describen en el mecanismo 'reflection' [ReflectionJAVA10] [ReflectionNET10] que se analiza en el capítulo 4, apartado 3.3:

- Métodos (nombre y firma (signature en inglés)).
- Propiedades (nombre y tipo).
- Eventos (nombre y firma (signature en inglés)).
- Metainformación (nombre del servicio, tipo del servicio, etc.).

Por lo tanto, la descripción unificada define los datos requeridos para la descripción y el consumo de los servicios.

La capa Bridges o 'puentes' se compone del conjunto de componentes que actúan como *gateways* hacia espacios de servicios estandarizados. Cada bridge se encarga de ofrecer una interfaz para el espacio de servicios correspondiente (esto es, UPnP,

WS), y para ello parte de la descripción unificada del servicio. Por ejemplo, un bridge de UPnP es un puente hacia un dispositivo UPnP que ofrece los servicios de este dispositivo basándose en la descripción unificada. De la misma manera, un bridge hacia Servicios Web debería instanciar un Servicio Web basándose en la misma descripción unificada. La capa Bridges se encargará de publicar los servicios con la tecnología que corresponda, esto es, con el protocolo de descubrimiento de servicios.

Todos los bridges de esta capa son completamente independientes de las tecnologías subyacentes y esto garantiza la escalabilidad y la extensibilidad del sistema. Se pueden incorporar nuevas tecnologías de dispositivos simplemente añadiendo nuevos drivers. De la misma manera, se pueden integrar nuevas tecnologías de servicios por medio de nuevos bridges. Es más, se debe destacar que estas extensiones no influirán en el funcionamiento del entorno en absoluto.

Destacar que el diseño de la aplicación se ha basado en los siguientes principios: primero, una aplicación es cliente de un conjunto de servicios y para ello hace uso de un protocolo de descubrimiento de servicios y de interacción específicos; segundo, el desarrollador de aplicaciones elige la tecnología de servicios a utilizar; y tercero, las aplicaciones hacen uso de los servicios disponibles, teniendo en cuenta su dinamismo. Un servicio puede abandonar la red y la aplicación debería ser capaz de buscar uno similar, si lo hay, y dinámicamente registrar y hacer uso del nuevo servicio.

Uno de los principales objetivos del *middleware* propuesto, como se ha mencionado, es permitir a los programadores seleccionar una tecnología de servicios que mejor se ajuste a sus necesidades independientemente del propio utilizado por los dispositivos. Esto significa que los servicios nativos se tienen que mapear con otros de más alto nivel. Como se ha mencionado previamente, esto se consigue en dos pasos y el Modelo Unificado de Servicios define el nivel intermedio entre ellos.

A continuación se muestra de forma más detallada el Modelo Unificado de Servicios.

3.3.1 Modelo unificado de servicios

El modelo unificado de servicios presenta un modelo para un acceso común a los servicios ofrecidos por diversos dispositivos. Así pues, cada uno de los servicios nativos de los dispositivos se asociará con una instancia de un servicio descrito en el modelo unificado. La figura 3.2 muestra el diagrama de clases para este modelo.

Comenzando desde la parte superior de la figura 3.2, vemos la clase ‘Service’ o *servicio*. Un servicio se puede describir por medio de un conjunto de métodos, propiedades y eventos. Por lo tanto, un servicio puede ser modelado funcionalmente como una colección de métodos, propiedades y eventos. Así, este modelo abarca cualquier servicio descrito según el paradigma de la programación orientada a objetos. Por otro lado, los parámetros (*parameter*), valores de retorno (*return value*) y los tipos (*type*) asociados son necesarios y se establecen como parte del contrato, WSDL por ejemplo, que son necesarios para que dos sistemas o aplicaciones se entiendan e intercambien información sobre un formato de datos conocido.

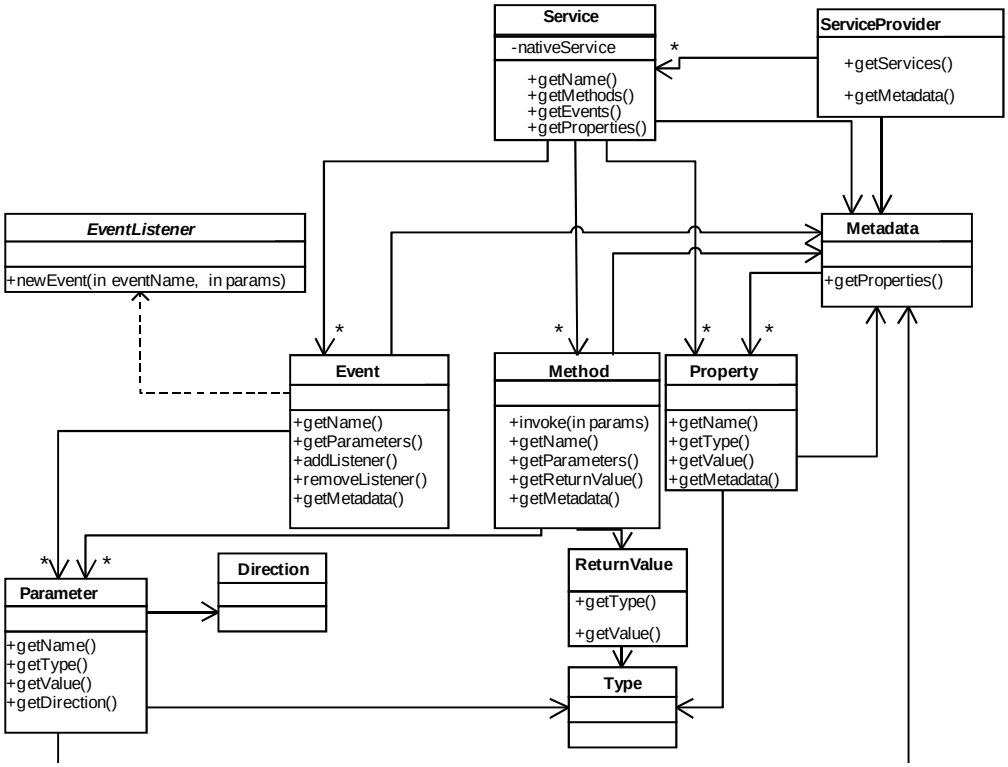


Fig. 3.2 Diagrama de clases del Modelo de Servicios

Las propiedades no sintácticas de algunos de los elementos mencionados con anterioridad (el proveedor de servicios (*ServiceProvider*), la localización, el fabricante...) son también recogidos y modelados como metadata. Se ha propuesto un modelo tal que sea lo suficientemente genérico como para recopilar toda la información que se pueda extraer de un servicio nativo.

3.3.2 Modelo del sistema

Un modelo del sistema ayuda a entender la funcionalidad del sistema y los flujos de datos existentes entre las aplicaciones y los dispositivos. La figura 3.3 muestra el modelo del sistema al completo, que se compone de un conjunto de dispositivos heterogéneos, el *middleware* descrito previamente y un conjunto de aplicaciones que consumen los servicios que ofrece el *middleware*.

En general, las aplicaciones ubicuas usan un protocolo de descubrimiento específico (por ejemplo, UPnP, WS). Una aplicación que se basa en UPnP encontrará los servicios de los dispositivos UPnP que haya en la red y hará uso de los mismos. De la misma manera, una aplicación basada en Servicios Web podrá hacer uso de los Servicios Web disponibles. Por medio del *middleware* propuesto las dos aplicaciones podrán acceder a los mismos servicios, pero mediante distintos protocolos de descubrimiento e interacción. Cada aplicación hará uso del protocolo utilizado. Esto se consigue mediante la instanciación de servicios en los correspondientes espacios de servicios.

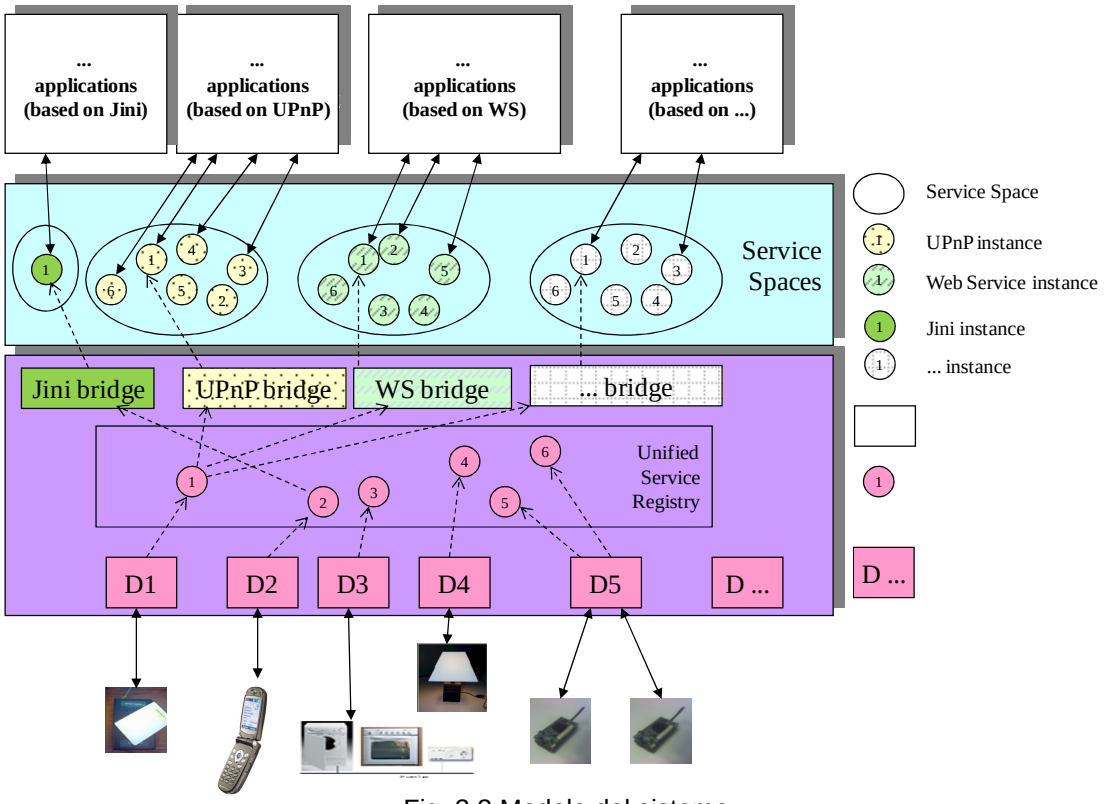


Fig. 3.3 Modelo del sistema

El *middleware* propuesto funciona de la siguiente manera. En la figura 3.3, se puede ver una lámpara con tecnología T4. El D4 es un driver hacia la tecnología T4, esto es, permite el acceso a un servicio ofrecido por el dispositivo lámpara (por ejemplo, apagar, encender, controlar la intensidad de la luz, y puede extraer toda la información sobre el servicio. Por lo tanto, se va a instanciar un objeto de tipo Servicio Unificado que contenga toda la información de los servicios nativos ofrecidos por la lámpara. Los servicios de este objeto posteriormente se registrarán en el Registro de Servicios Unificados (*Unified Service Registry*). A continuación, los bridges asociados a cada protocolo de descubrimiento se suscribirán a estos nuevos servicios.

Merece la pena destacar que los bridges no dependen de la tecnología subyacente, sólo hacen uso de los objetos de tipo Servicio Unificado que están registrados en el Registro de Servicios Unificados. Así, se crean instancias de los servicios en el espacio de servicios correspondiente. Distintos bridges pueden coexistir, cada uno creando su propio espacio de servicios.

3.3.3 Arquitectura interna

En la figura 3.4 se ofrece una vista más detallada de la solución *middleware* propuesta. La capa 'Service discovery' de dicha figura se encarga de descubrir los servicios de los dispositivos disponibles con su tecnología correspondiente, ofreciendo los servicios de estos a capas superiores.

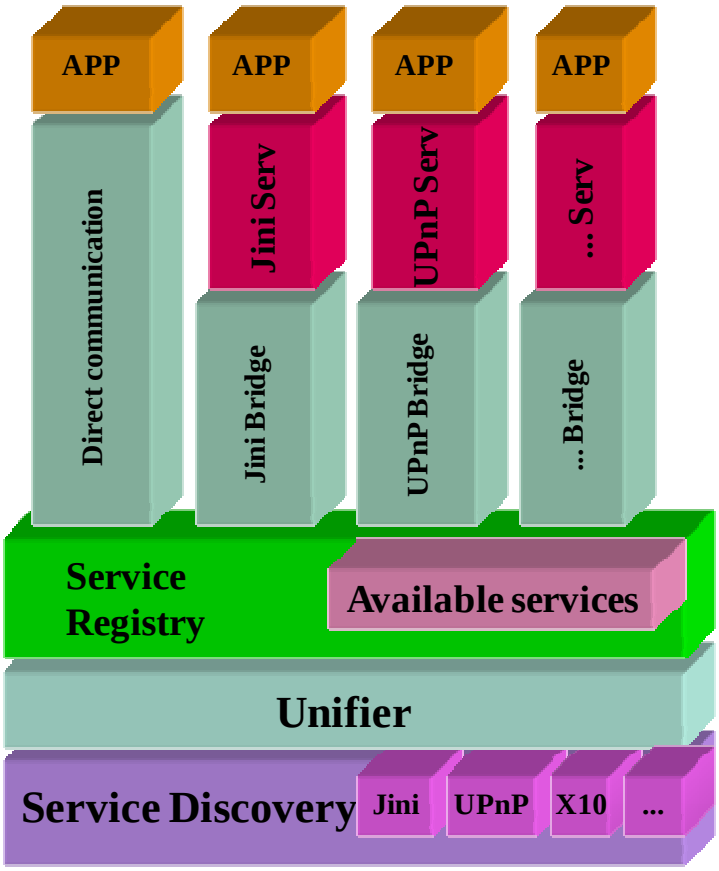


Fig. 3.4 Vista detallada del *middleware*

Este descubrimiento se realiza según el protocolo propietario con el que el dispositivo fue concebido. La capa 'Unidied Model layer' de la figura 3.1 se descompone en la capa 'Unifier' y la capa 'Service Registry' en la figura 3.4. La capa 'Unifier' es responsable de la creación de una descripción unificada de cada uno de los dispositivos descubiertos. Por otro lado, el 'Service Registry' se corresponde con la capa 'Unified Modfel Layer' de la figura 3.1, al igual que los diferentes 'Bridges' (Jini, UPnP…) se corresponden con la capa Bridges de la figura 3.1. Por último, 'Jini Service' se refiere al espacio de servicios JINI, con el que interactuará la aplicación que quiera utilizar comunicación tipo Jini para comunicarse con los dispositivos.

La descripción de un servicio estará formada por la información publicada en una dirección '*multicast'* y su correspondiente puerto UDP/TCP, tal y como se muestra en un ejemplo en la figura 3.5.

```
PUT http://localhost:12225/Device/Locator/Information_Service/location HTTP/1.1
Host: 224.100.0.1:8888
Content-Length: 16
Content-Type: application/x-www-form-urlencoded

Service=Location
User=McQueen
```

Fig. 3.5 Captura mensajes para envío '*multicast'*

Las descripciones unificadas de los servicios se almacenan en el 'Service Registry'. A continuación, se explicará con un ejemplo gráfico las interacciones entre las capas que hemos descrito anteriormente, comenzando desde el más bajo nivel.

3.3.3.1 Descubrimiento y autoconfiguración

Algunos autores proponen la integración y adaptación del *middleware* a cada tipo de protocolo que se utiliza [Allard+03] [Sameh+04], mientras que otros han desarrollado sus *middleware* integrando los protocolos conocidos de manera que no se puedan utilizar otros [Salvador+05]. En esta tesis se ha seguido la primera aproximación, no restringiendo el uso de los protocolos, dejando en manos del desarrollador y de las aplicaciones la elección del más conveniente en cada momento. Concretamente en esta tesis se han desarrollado bridges para UPnP y Jini.

3.3.3.2 Mapeo entre las capas Drivers y Unifier

El desarrollador se encargará de describir en un modelo unificado cada uno de los servicios de los dispositivos. Una vez hecha la descripción unificada, será enlazada con la capa *Unifier*. Es decir, si llegado el momento es necesario comunicarse con un dispositivo o familia de dispositivos que utilizan un protocolo concreto y propietario no soportado por el sistema, no habrá más remedio que generar el *wrapper* correspondiente vía las APIs proporcionadas por el fabricante. Como cada API es totalmente dependiente de cada implementación propietaria del dispositivo, la obtención de los diferentes métodos, propiedades, eventos y metainformación para mapearlos con la estructura *Unifier* deberá ser generado *ad-hoc*.

3.3.3.3 Mapeo entre las capas Unifier y Bridges

Una vez que se dispone de una descripción unificada de los servicios de los dispositivos y cada uno de ellos se ha registrado en 'Service Registry', la invocación de los servicios a través de los diferentes bridges dependerá de las necesidades particulares de las aplicaciones.

A parte de los bridges UPnP y Jini, si una aplicación utilizara DLNA, habría que incorporar un nuevo bridge para dicha tecnología. Una vez creado dicho bridge, y gracias al *middleware*, se crearía un nuevo espacio de servicios DLNA. En este sentido, el *middleware* está vivo, ya que se puede adecuar añadiendo o quitando módulos para cubrir las necesidades existentes en cada ubicación o momento.

Tal y como se ha mencionado anteriormente existen protocolos de descubrimiento como Jini, SLP, DLNA o Bluetooth SDP. Ninguno de ellos domina el mercado, por lo que si el *middleware* desarrollado es capaz de descubrir todo tipo de dispositivos y queremos que sea utilizable en cualquier tipo de entorno, deberá proveer puentes 'Bridges' para cada protocolo con el que necesitemos interactuar para habilitar el paso de mensajes entre las diferentes capas del *middleware*.

3.3.3.4 Ciclo de vida de un servicio interoperable

La figura 3.6 muestra un diagrama de secuencia de actividades que ocurren en el *middleware* cuando un nuevo dispositivo (con sus correspondientes servicios) aparece en la red. En el diagrama de secuencia, se muestran los componentes del *middleware* como líneas de vida en vertical y sus interacciones en el tiempo están representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Así, este diagrama muestra qué componentes del *middleware* se comunican entre sí y qué mensajes existen entre ellos.

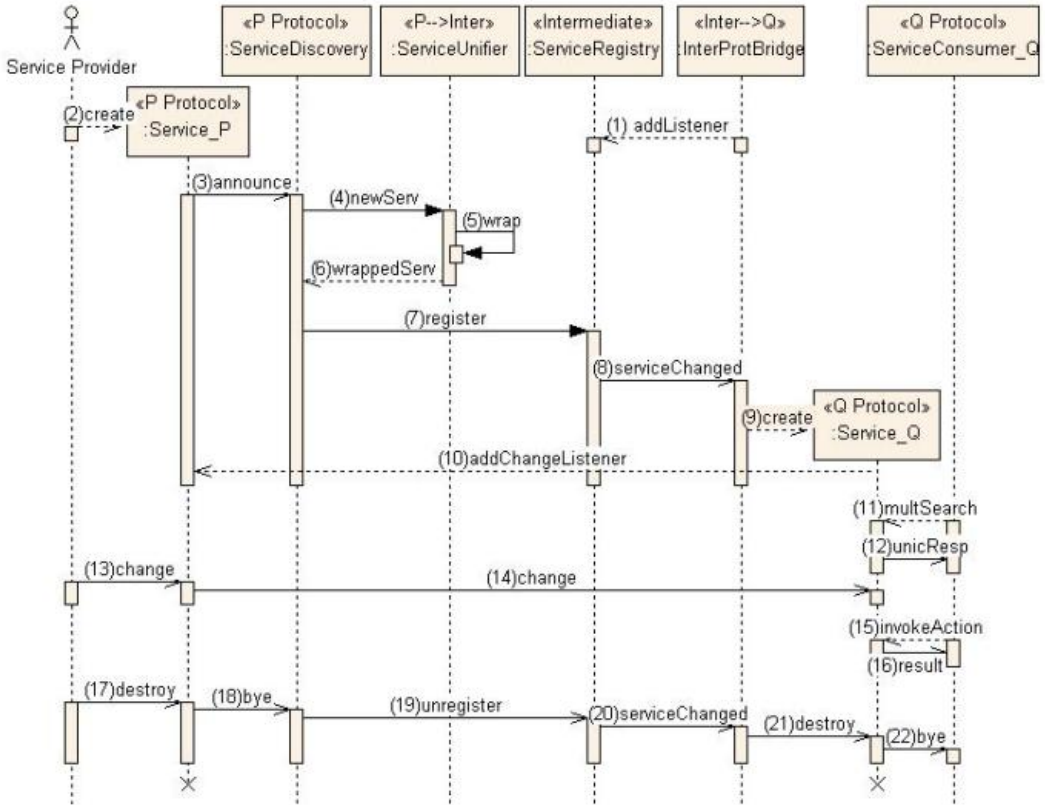


Fig. 3.6 Diagrama de secuencia cuando se descubre un nuevo dispositivo/servicio en la red

La secuencia de lo que ocurre en el *middleware* es:

I. Los 'bridges' entre protocolos de descubrimiento de servicios se suscriben (se dan de alta) en el 'Service Registry' para que puedan recibir notificaciones de nuevas altas y bajas de dispositivos o servicios (mensaje (1) en la figura 3.6).

II. Cuando un dispositivo se anuncia y envía el mensaje correspondiente a la red utilizando el protocolo P (mensaje (2)), la capa 'Service Discovery' recibe el mensaje con las descripciones de los nuevos servicios disponibles (mensaje (3)).

III. En la capa 'Unifier', a través del formato de intercambio de información del protocolo P, se obtiene la descripción de los servicios (mensajes (5) y (6)) y se registran en el 'Service Registry' (mensaje (7)).

IV. A continuación, el 'Service Registry' notifica a sus suscriptores (los diferentes 'bridges' inter protocolos) que un nuevo servicio ha sido registrado (mensaje (8)). Cada 'Bridge' lanzará un servicio con formato de otro protocolo Q (mensaje (9)) y se suscribirá al servicio original (mensaje (10)).

V. De esta manera, cualquier cambio que ocurra en el servicio original será automáticamente notificado a su servicio asociado en otro protocolo (mensajes (13) y (14)).

VI. Una vez que el servicio descrito en el protocolo Q está disponible en la red, estará en disposición de responder a cualquier aplicación que realice una búsqueda multicast para ese protocolo (Q) (mensajes (11) y (12)).

VII. Desde ese momento, esas aplicaciones serán capaces de invocar las acciones descritas para ese servicio (mensajes (15) y (16)).

Cuando un servicio original es eliminado, porque el dispositivo se ha apagado por ejemplo, el 'service discovery' correspondiente al protocolo con el que se comunica con el dispositivo recibe un mensaje (18) de fin de servicio o no disponible. Con ello se produce una serie de mensajes para el desregistro. Se desregistran en el 'service Registry' (mensaje (19)) y en los diferentes 'bridges' inter protocolos (mensaje (20)), de tal manera que a su vez se desregistran de los servicios que previamente habían lanzado (mensaje (21)), y finalmente notifican a las aplicaciones que están haciendo uso del servicio de la desaparición del servicio (mensaje (22)).

3.4 Evaluación del middleware

Como se ha mencionado anteriormente, el objetivo final es presentar los servicios de los dispositivos hardware como servicios de software estándares y genéricos a las aplicaciones de alto nivel. Para evaluar y estudiar el funcionamiento del *middleware*, se han diseñado e implementado dos aplicaciones ubicuas. En el primer caso se pretende controlar la luminosidad en una habitación dependiendo de la ubicación de cierto usuario. En el segundo,, se ha realizado una aplicación para la detección de caídas con la utilización de diferentes acelerómetros. En ambos casos, el *middleware* proveerá a la aplicación de fácil acceso y control de los distintos servicios de los dispositivos.

3.4.1 Caso práctico 1

Los dispositivos que intervienen son: sensores de luminosidad, lámparas 'domóticas' y sensores de localización de usuarios. Se han utilizado localizadores basados en la tecnología RFID que nos dan una localización discreta, es decir, si una persona está o no está en una ubicación. También se han empleado sensores MicaZ con el sistema operativo TinyOs para la monitorización de la luminosidad y dos lámparas, una de ellas conectada a un puerto serie RS232 y la otra al Bus Domótico Fagor (BDF) [Fagor10] vía comunicación *powerline*. Los sensores MicaZ informan del nivel actual

de luz, mientras que ambas lámparas pueden ser apagadas/encendidas remotamente de acuerdo al protocolo específico.

La aplicación enciende las lámparas siempre que el usuario preestablecido se encuentre en la habitación y no haya suficiente luminosidad. En cualquier otro caso, las lámparas son apagadas. Desde el punto de vista de la aplicación ubicua, lo único que necesita saber es que existen unas lámparas que se pueden encender y apagar, es decir, que las lámparas ofrecen un servicio de encendido y apagado. Estos servicios de encendido y apagado serán invocados por las distintas aplicaciones de alto nivel de manera estándar gracias al *middleware*. Precisamente, las aplicaciones ubicuas no tienen que conocer la naturaleza o la tecnología base de los servicios de los dispositivos. En este caso, da igual que una lámpara use una interfaz de comunicación RS-232 o que esté conectada mediante el BDF; la aplicación mediante el uso del *middleware* controlará ambas lámparas de la misma forma, haciendo uso de los servicios disponibles. Desde el punto de vista de las aplicaciones, todos los dispositivos se verán como proveedores de servicios estándar. En esta aplicación se ha utilizado UPnP como servicio de descubrimiento e interacción con los dispositivos. Es decir, se ha desarrollado un punto de control UPnP para descubrir e interactuar con cualquier dispositivo UPnP en la red. Así, el *middleware* propuesto incorpora un bridge UPnP para que todos los servicios disponibles en dicha red sean accesibles vía UPnP. Los dispositivos UPnP se instancian mediante el bridge UPnP y los servicios de localización de usuarios, sensores de luminosidad y lámparas serán descubiertos y usados por la aplicación. Esto es, una vez los descubren, la aplicación podrá invocar acciones y suscribirse a eventos.

3.4.2 Caso práctico 2

Mediante la utilización de dispositivos como detectores de presencia, sensores MicaZ con modulo de acelerómetros, teléfono móvil, bocina… se ha realizado una aplicación para la detección de caídas. La aplicación principal detecta la posible caída al recibir la información desde los acelerómetros. A continuación, y dependiendo del lugar en el que se encuentre, información que provee el detector de presencia, se evaluará si es pertinente hacer sonar la bocina o el avisador. Por otro lado, también se envía un mensaje de texto a las personas prefijadas. Las aplicaciones se han creado para utilizar los bridges UPnP. La aplicación principal está desarrollada en VB .NET, y la aplicación de envío de mensajes de texto en Java. Gracias al *middleware* que facilita la interconexión de aplicaciones y dispositivos, habilitando la suscripción a los eventos pertinentes de cada dispositivo, con la consiguiente lógica de aplicación y secuencia de acciones a realizar, se ha generado un sistema para la detección de caídas.

3.5 Evolución middleware – CAHIM

Con el paso del tiempo y viendo la evolución y la tipología de escenarios ubicuos en los que debía ser utilizado el *middleware*, fueron detectadas nuevas necesidades. Por

ejemplo, la gestión de la información del estado de los servicios y datos provenientes de los diferentes sensores y dispositivos heterogéneos móviles se convierte en laboriosa para el programador o usuario final en un entorno distribuido en el que los dispositivos y servicios aparecen y desaparecen, por lo que se hace necesario dotar al sistema, al *middleware* en este caso, de mecanismos que faciliten la gestión del entorno (por ejemplo, detectar y gestionar dispositivos que aparecen/desaparecen en la red). Además, otra característica que el *middleware* podría gestionar es la configuración de las aplicaciones en función de los servicios o recursos disponibles en la red.

3.5.1 Motivación

Las aplicaciones ubicuas necesitan ser conscientes de los cambios (recursos de red o dispositivos que aparecen y desaparecen, nuevos dispositivos, configuraciones que cambian en función de los servicios disponibles) y de las necesidades que el usuario tiene. Las aplicaciones ubicuas necesitan adaptarse a dichos cambios, muchas veces incluso anticipándose a ellos [Abowd+00]. Ser consciente del contexto que rodea al usuario es de vital importancia para proveer de aplicaciones fiables y con buen rendimiento. Para ello, surge la necesidad de dotar al *middleware* de mecanismos para que provean un procesamiento inteligente de la información recibida, y de servicios fiables entre otros garantizando siempre la interoperabilidad entre dispositivos y aplicaciones [Uribarren+08].

Así pues, un *middleware* encargado de facilitar la comunicación entre dispositivos heterogéneos y aplicaciones debería también encargarse de recolectar información de contexto y, finalmente, de preprocesar toda la información generando nueva información adicional útil para lograr aplicaciones ubicuas sensibles al contexto. Todo ello debería ser transparente para las aplicaciones, siendo estas las que se adaptan en función de lo que en cada momento el *middleware* está ofreciendo. Adoptar estrategias que se adapten a los continuos cambios del entorno y necesidades del usuario es esencial. Es en este sentido donde se pretende extender la funcionalidad del *middleware*, proveyendo los mecanismos para almacenar y procesar la información referida al contexto y generar nueva información a partir de ella. La parte referente a la 'comunicación transparente' entre dispositivos y aplicaciones ya ha sido abordada en el apartado 3.3.2.

3.5.2 Nuevos retos en el middleware

La visión de Weiser respecto de la computación ubicua es aquella en que los sistemas y dispositivos con capacidad de procesamiento se integran de manera transparente en la vida cotidiana de cualquier persona [Weiser91]. Alrededor de esta idea, se ha hecho más palpable [López-de-Ipiña+06].la dificultad de desarrollar nuevas aplicaciones que se adapten al entorno, que se mantengan vivas y funcionando cuando la gente se mueva de una ubicación a otra, y que encima no necesiten de mantenimiento (zero administration),

Un entorno ubicuo debería ser consciente de la presencia de usuarios y ser sensible a los deseos, hábitos y emociones de los mismos [Ferscha03, Saha+03]. Para satisfacer estos requisitos, las aplicaciones ubicuas deberían proveer: acceso ubicuo, ser conscientes del entorno que les rodea, inteligencia e interacción transparente con el usuario [Ferscha03]. En otras palabras, el *middleware* facilita a las aplicaciones tareas como la gestión de protocolos, gestión de fallos de comunicación o problemas de heterogeneidad [Saha+03, Da Costa+08].

Resumiendo, el *middleware* debería ser responsable de la:

- *Configurabilidad*: gestionar las configuraciones de las aplicaciones en función de los servicios o recursos de red disponibles.
- *Adaptabilidad*: debería detectar y gestionar los cambios en el contexto. Por ejemplo, detectando nuevos dispositivos y/o servicios, movimientos del usuario, aunque también creemos que cierta gestión de adaptabilidad debería ser llevada a cabo por las propias aplicaciones.
- *Heterogeneidad*: debido a la gran variedad de dispositivos, el *middleware* debería soportar y escalarse a todos ellos.
- *Interoperabilidad*: facilitar la comunicación y el trabajo en grupo de una red heterogénea de dispositivos en la consecución del objetivo marcado por la aplicación.

Señalar que nuestra primera aproximación de *middleware* expuesta en el apartado 3.3.3 daba respuesta a las características de interoperabilidad y heterogeneidad. A continuación, se estudiará la adaptabilidad y configurabilidad que deberá proveer el *middleware*.

Dado que el *middleware* será responsable de las propiedades anteriores, a partir de ahora se denominará CAHIM (*Middleware* Configurable, Adaptable, Heterogéneo e Interoperable).

3.5.3 Configurabilidad y adaptabilidad

El sistema que se presenta a continuación [Uribarren+06b] pretende servir como base para aplicaciones ubicuas que se ejecutarán en un entorno móvil. Muchas de las aproximaciones plantean el intercambio de información entre dispositivos en términos generales, de tal forma que se recolecta información referente a los dispositivos y servicios, se transforma, se adapta y se distribuye entre las aplicaciones interesadas. En nuestro caso, el objetivo principal es intercambiar la información de contexto, y proveer de los mecanismos necesarios para que los servicios, ejecutándose en el mismo dispositivo o en diferentes, sean conscientes de los recursos del entorno, de tal forma que se puedan adaptar y proveer de servicios a las aplicaciones en las capas superiores.

Para lograr la adaptación, coordinación y autoconfiguración de servicios, una nueva capa de control ha sido añadida a nuestra primera iteración de la solución *middleware*

(Fig. 3.7). La capa de control está compuesta de tres componentes 'Coordinator', 'User profiling' y 'Context information'.

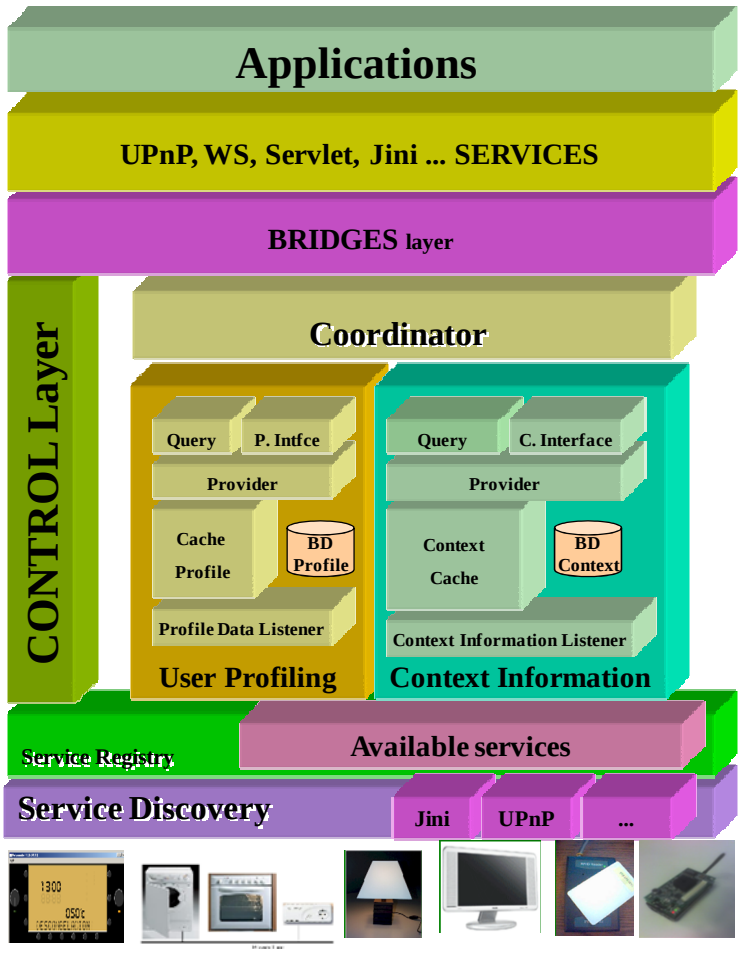


Fig. 3.7 Capa de control

La capa de control es la responsable de mantener el estado actual del *middleware* en-línea y actualizado, y de proveer los mecanismos para posibilitar la reconfiguración de acuerdo con las reglas establecidas y de la información del estado actual y pasada almacenada en el sistema. Combinando las informaciones de contexto generales del estado de una ubicación con la información referida a los perfiles de uso de cada usuario, la capa de control será capaz de componer y recrear un entorno siguiendo las directrices marcadas por las aplicaciones superiores.

El componente '*context information*' se encarga de mantener el estado actualizado de los dispositivos y servicios en el middleware. Además, se encarga de almacenar los cambios de estado para poder dar respuesta a las peticiones que puedan llegar desde las aplicaciones superiores.

El componente '*user profiling'* es similar al '*context information'*, pero la información que trata y almacena es la referente a los usuarios, es decir, sus preferencias, deseos o necesidades.

El componente '*coordinator'* juega un rol muy importante en el *middleware*. Se trata de un componente supervisor con una serie de reglas, reglas ECA y objetivos asociados. Su objetivo principal es la coordinación y orquestación de dispositivos y servicios para satisfacer los objetivos de las aplicaciones superiores. La información del contexto y de usuario será enviada y utilizada por el coordinador para tomar decisiones y realizar las acciones pertinentes en caso de conflictos o nuevas situaciones. En el capítulo 5 Configurabilidad se explica con detalle las partes de las que consta este componente coordinador y el funcionamiento dentro de una red de sensores y dispositivos. En la misma se analiza el funcionamiento del '*Motor de Reglas'* ECA propuesto que realiza las labores de '*coordinator'.* Por ejemplo, si una aplicación solicita encender las luces de una habitación, el coordinador localiza los dispositivos y servicios que pueden proveer luz en la habitación para poder encenderlos. Todo dispositivo que contenga el servicio 'luz' será invocado por el coordinador. Por otro lado, y siguiendo las preferencias del usuario y analizando el nivel de luminosidad preferido por el usuario, el coordinador chequea si el nivel obtenido activando todos los dispositivos con el servicio requerido coincide con su valor preferente. Si no es así, realizará otra serie de acciones suplementarias como por ejemplo subir las persianas antes de responder negativamente a la petición de la aplicación superior. Como se muestra en el ejemplo anterior, algunas de estas reglas del coordinador darán respuesta a los servicios requeridos utilizando vías alternativas, siempre en aras a proveer de la respuesta adecuada a las aplicaciones superiores. La especificación de cómo se crean las reglas utilizadas está detallada en el capítulo 6.

El despliegue del componente '*coordinator'* en el *middleware* puede ser distribuido y fragmentado como se requiera en cada entorno de aplicación, distribuyendo los subcomponentes en diferentes dispositivos involucrados. Es decir, lo que más adelante en el capítulo 5 de esta tesis denominaremos '*Motor de Reglas*' podrá ser distribuido siguiendo una arquitectura orquestada o en coreografía, ajustándose a las necesidades de cada escenario de aplicación.

3.6 Conclusiones

Acabamos de presentar el diseño de un *middleware* programable de alto nivel y la infraestructura necesaria para desarrollar aplicaciones y servicios ubicuos, tal que:

I. Ofrece un mecanismo uniforme, para que a través de mapeos concretos, los dispositivos puedan ser utilizados desde cualquier aplicación con cualquier protocolo de interoperabilidad existente de forma natural y transparente.

II. Ofrece un mecanismo de virtualización de los dispositivos ofreciendo flexibilidad y libertad a la hora de programar y generar nuevas aplicaciones, siendo menos intrusiva y agresiva la utilización de nuevos dispositivos en entornos de desarrollo conocidos. Se superan las barreras de la *Heterogeneidad*, soportando la gran variedad de familias de dispositivos.

III. Flexibiliza y amplía el acceso y uso a los dispositivos, ya que un dispositivo puede llegar a ser utilizado o invocado utilizando diferentes protocolos simultáneamente.

IV. Fomenta la *Interoperabilidad* facilitando la comunicación y el trabajo en grupo de una red heterogénea de dispositivos en la consecución del objetivo marcado por la aplicación.

Con la utilización de los mecanismos explicados en este *middleware*, las aplicaciones son capaces de recibir y utilizar toda la información necesaria, tal y como a las aplicaciones les conviene. La utilización de este tipo de mecanismos en los entornos ubicuos ofrece la posibilidad de formular e idear sistemas que son capaces de hacer uso de los elementos del entorno de una manera generalizada independientemente del hardware y software utilizado.

Por otro lado, mediante los nuevos componentes añadidos en la evolución del middleware framework CAHIM, se ha dotado de un sistema flexible para ayudar en la generación de aplicaciones ubicuas. Esta flexibilidad hará que el *middleware* cumpla los objetivos que marcamos en el inicio de este capítulo, siendo un soporte para las aplicaciones en general en cuanto a:

V. *Configurabilidad*: proveyendo de los resortes necesarios para modificar las configuraciones de las aplicaciones en función de los servicios o recursos de red disponibles.

VI. *Adaptabilidad*: proveyendo mecanismos para detectar y gestionar los cambios en el entorno. Posibilitando la detección de nuevos dispositivos y/o servicios, errores o movimientos del usuario para posteriormente reaccionar siguiendo las preferencias establecidas por los propios usuarios.

Tal y como se ha comentado anteriormente, el middleware framework CAHIM es la base en la que se sustenta todo lo referente a la personalización de entornos ubicuos, al igual que todas la aplicaciones que tienen que ver con la interacción de los usuarios y los sistemas. En el capítulo 5 se analizan la Personalización y Configuración desde el punto de vista del usuario final, proponiendo el uso de cierta metodología de funcionamiento para abordar con éxito los escenarios del hogar u oficina que se plantean. De todas formas, hay aspectos importantes a desarrollar. La forma en que la información de comunicación es expresada y las necesidades de información entre las distintas aplicaciones en computación ubicua son aspectos a trabajar en el futuro, y escapan del ámbito de actuación del *middleware*. En este sentido hay que destacar que, aunque se están dando pasos en la generación y estandarización de categorías de dispositivos y los servicios ofrecidos, está costando que todos los fabricantes de dispositivos y bienes de equipo se pongan de acuerdo en este aspecto. La utilización de categorías comunes será vital para lograr una interoperabilidad integral entre sistemas y dispositivos, a la vez que retadora para la comunicad tanto científica como mas industrial. Para agilizar ese proceso de conversión existen tecnologías que están siendo estudiadas y aplicadas en la informática. La ontología es un concepto muy

recurrido en el campo de la representación del conocimiento, que hace referencia a formular un riguroso y exhaustivo esquema conceptual dentro de un dominio que facilite la comunicación y compartición de información. Los servicios anunciados deberán asumir, presumiblemente y en un futuro cercano, la tarea de indicar o crear su asociación en el esquema semántico.

Capítulo 4: Movilidad de aplicaciones

La necesidad del acceso y la utilización de los diferentes servicios independientemente de dónde se encuentre una persona lleva consigo una serie de necesidades implícitas. En este sentido, la necesidad de proveer al usuario de mecanismos que automáticamente detecten que ha habido un cambio de ubicación y se muevan las aplicaciones o servicios asociados es imprescindible. En este capítulo proponemos un *middleware*, que proveerá de los mecanismos necesarios para la instalación y ejecución remota de programas ejecutables en los distintos dispositivos. El objetivo es ofrecer al usuario la misma funcionalidad cualquiera que sea la ubicación, dispositivo en uso, u otra serie de parámetros del entorno que en un momento dado se estén utilizando. Todo de forma automática, transparente y ubicua para el usuario, instalando y ejecutando aplicaciones en la plataforma destino, siendo independientes los parámetros de configuración del código ejecutable utilizado.

Tal y como se ha explicado en el capítulo anterior de la tesis, la adaptación y configuración del entorno que le rodea al usuario en función del contexto y necesidades del usuario es vital. Por ello, en este capítulo de tesis y en el siguiente se van a analizar varias formas para llevarlo a cabo. Un resultado de la configuración de un entorno puede ser el variar los parámetros de funcionamiento de cierto dispositivo, es decir, encender o apagar la luz al entrar en una estancia, o cambiar de canal en el televisor automáticamente según las preferencias del usuario cuando el sistema detecta la entrada en el salón. Este tipo de configuración se tratará en profundidad en el capítulo 5 y 6. Por otro lado, la adaptación, personalización y configuración pueden necesitar que el sistema reconfigure total o parcialmente los componentes que se están ejecutando en los dispositivos. Para ello, será necesario proveer de mecanismos que permitan la movilidad, incluyendo la carga de aplicaciones y configuraciones personales, su ejecución, al igual que parar su ejecución y liberar espacio desinstalándolas. Este último modo de funcionamiento que involucra la carga, descarga y configuración de aplicaciones será el objeto de este capítulo. La inclusión o suma de las diferentes propiedades que se analizan en cada capítulo dará como resultado un *middleware* capaz de responder a las necesidades que se plantearon al comienzo de la introducción como objetivos de esta tesis.

En esencia, la movilidad y configuración dinámica de componentes serán propuestas como mecanismos innovadores para lograr la deseada configuración y personalización de los espacios laborales y del hogar.

4.1 Introducción

Actualmente, existen en el mercado multitud de dispositivos (por ejemplo, PCs, PDAs, móviles, ultra PCs) que ejecutan diferentes plataformas no compatibles entre sí. La adaptación de aplicaciones para que puedan ser ejecutadas en cualquier dispositivo puede llegar a ser una tarea muy tediosa y a veces imposible, ya que las restricciones de hardware y software pueden llegar a ser insuperables. Incluso los nuevos requisitos de interoperabilidad, portabilidad y accesibilidad introducen y enfatizan el necesario carácter dinámico que una aplicación debe tener a la hora de formar parte de una u otra infraestructura de manera transparente. El objetivo es lograr que las aplicaciones puedan ser accesibles o utilizables en diferentes plataformas y entornos de trabajo en los que la movilidad del usuario debe ser cubierta y satisfecha. Este tipo de capacidad o mecanismo es esencial en entornos de computación ubicua para soportar aplicaciones sensibles al contexto [Dey01], que necesiten intercambiar información adaptándose entre los diferentes dispositivos con sistema operativo diferente [Weiser91, Sawhney+00].

Las nuevas aplicaciones deberán ser más conscientes (sensibles) al contexto que les rodea [Dey01], por lo que deberán proveer servicios inteligentes dirigiendo y reduciendo la interacción entre persona y aplicación. El contexto de una aplicación se puede entender como “cualquier información que se puede utilizar para definir la situación de una entidad, donde una entidad es una persona, lugar u objeto que es considerado relevante para la interacción entre un usuario y una aplicación, incluyendo al usuario y a la propia aplicación”, tal y como se define en [Dey+01].

En este capítulo también se van a explicar y analizar los problemas surgidos a la hora de adaptar las aplicaciones a las diferentes plataformas, y se propondrá la infraestructura necesaria para soportar la movilidad y reconfiguración de las aplicaciones en entornos heterogéneos. Las soluciones propuestas se usarán en aplicaciones sensibles al contexto como son la automatización del hogar, de una empresa, etc.

En un entorno en el que cada día más elementos están interconectados, las personas somos más dependientes de dispositivos, como un ordenador, portátil, PDA o teléfono móvil. Estos dispositivos son tan útiles para nosotros, porque contienen una serie de aplicaciones instaladas que nos ofrecen unos determinados servicios y que normalmente están asociadas a uno o varios servidores. La movilidad de aplicaciones puede encontrarse con barreras tales como: que las aplicaciones no están instaladas en todos los dispositivos, puede existir diferencia en cuanto al sistema operativo, no todos los dispositivos cuentan con la misma capacidad de procesamiento, resolución de pantalla, etc. En muchos casos esta serie de limitaciones se convierte en insalvable para el usuario a la hora de realizar las tareas que quiere, dónde quiere y cuándo quiere.

La movilidad de aplicaciones y la posibilidad de acceso a la información no siempre van de la mano. El acceso a programas y datos en una nueva ubicación puede

requerir la instalación de nuevos programas para seguir teniendo la misma funcionalidad, o incluso puede que no sea posible debido a un sistema operativo diferente. En este sentido, se hace necesario algún tipo de mecanismo para que las aplicaciones sigan funcionando sea cual sea el sistema operativo o dispositivo cumpliendo las expectativas que en cada momento son demandadas por los usuarios.

Existe una serie de requisitos que definen la forma de interactuar en los sistemas ubicuos, que definen lo que se debe cumplir, establecen los retos tecnológicos a superar, la infraestructura necesaria y la manera en la que los componentes software y las interfaces de usuario se interrelacionan. Los componentes que juegan este papel fundamental se describen en los cuatro apartados siguientes y son: dispositivos, componentes software, usuarios e interfaces de usuario.

4.1.1 Dispositivos

En un entorno ubicuo son dos las características de los dispositivos que hay que tener en cuenta: heterogeneidad y movilidad.

La heterogeneidad de dispositivos requerirá de infraestructuras que den soporte a la integración de manera coherente e interacción arbitraria entre los mismos. Los dispositivos pueden ser desde PCs, PDAs, teléfonos móviles hasta sensores, actuadores, enchufes inteligentes y dispositivos embebidos, entre otros. Esta diversidad de dispositivos no desaparecerá en el futuro, sino que además aumentará. Serán los dispositivos los que nos proporcionen información del mundo físico trasladándolo al mundo digital/virtual. Este desafío y su solución se han tratado en el capítulo 3.

4.1.2 Componentes software

Básicamente, las características de las aplicaciones en una infraestructura ubicua han de ser la adaptabilidad[3], movilidad, interoperabilidad y sensibilidad al contexto, facilitando la generación y distribución de componentes software. También han de contar con mecanismos de descubrimiento de estos componentes.

El middleware debe garantizar la movilidad y distribución de componentes software, con la posibilidad de que los componentes se muevan en la misma máquina, o a diferentes máquinas, ubicadas a escasos metros o a cientos de kilómetros. Sin olvidar

[3] Adaptabilidad y flexibilidad son sinónimos según el 'standard glossary of software engineering terminology' [IEEE Glossary90]. Ambos se definen como la facilidad con que un sistema o componente puede ser modificado para su uso en aplicaciones o entornos distintos de aquellos para los que fue diseñado específicamente. También como la capacidad de los sistemas software para soportar los cambios en el entorno en el que se ejecutan.

que la transición de los componentes, paso de datos o la sincronización se deben realizar de forma transparente para el usuario.

La invisibilidad de los dispositivos, característica de la computación ubicua, vendrá dada por la reducción de la interacción entre las aplicaciones y las personas, reemplazando esta interacción con el conocimiento que adquiera el sistema del contexto. Los componentes software sensibles al contexto adquirirán información del entorno (por ejemplo, a través de sensores) como son la localización del usuario, la proximidad de objetos, hora del día, temperatura y humedad. Por ello, el contexto puede llegar a ser cualquier información que puede valer para describir la situación de una entidad.

En cuanto a la interoperabilidad, hoy en día, los desarrolladores de aplicaciones utilizan diversos lenguajes de programación y entornos de desarrollo, es por ello que una infraestructura ubicua deberá soportar esta diversidad. Por lo tanto, la integración de distintos componentes software residentes en entornos de desarrollo variopintos ha de ser esencial en este tipo de sistemas.

La movilidad de usuarios (y con ello la movilidad de dispositivos) añade varios retos desde la gestión de las conexiones, hasta la movilidad de aplicaciones. Un usuario al desplazarse de un lugar a otro va a necesitar muchas veces que la información se replique, esto es, que se carguen y ejecuten programas en otro dispositivo manteniendo las mismas configuraciones actuales, o incluso que se creen nuevas configuraciones dependiendo del contexto actual. Aunque los protocolos de red resuelvan ciertos problemas de conexión, también va a ser necesario proveer de mecanismos que garanticen la movilidad de la aplicación en el ámbito. A lo largo de este capítulo de tesis y más concretamente en el apartado 4.2, se abordará más en profundidad este aspecto fundamental para la computación ubicua.

La movilidad de una aplicación implica la consideración y realización de varias tareas en paralelo. La adaptación de la interfaz de usuario, configuración de la misma y garantizar su funcionamiento independientemente del destino en el que se ejecuta son tareas básicas e imprescindibles que hay que abordar en este tipo de sistemas. De alguna forma, se está tratando de abordar el problema de interoperabilidad de aplicaciones y plataformas conservando la configuración de la aplicación.

La movilidad de componentes software también está relacionada con el descubrimiento de dichos componentes y su distribución. En cuanto al desarrollo y distribución de componentes, señalar que para un rápido desarrollo de componentes, el *middleware* debe aislar al programador de todo lo referente al descubrimiento de recursos, métodos de comunicación y distribución. El *middleware* debería proveer de interfaces (APIs) simples pero potentes, bien definidas, separando claramente funciones propiamente del *middleware* y del componente. De esta manera, también se hace hincapié en la reusabilidad y portabilidad. El *middleware* debería dar soporte a las distintas plataformas y proveer de APIs bien definidas entre ellas, permitiendo la movilidad y reusabilidad de los componentes en las distintas plataformas.

4.1.3 Usuarios

Los usuarios en un entorno ubicuo son considerados móviles, con posibilidad de iniciar sesión en cualquier dispositivo. La infraestructura debería respetar y mantener la configuración existente y tener en cuenta el nuevo contexto originado por la movilidad. Por ejemplo, si el usuario está conduciendo, el sistema debería canalizar todas las comunicaciones de voz a través del servicio de manos libres. La identificación de las necesidades en un instante dado debería realizarse automáticamente por el sistema, siendo transparentes para el usuario, pudiendo incluso instalar aplicaciones y mover el contenido de las mismas para seguir dando la funcionalidad requerida por el usuario.

4.1.4 Interfaces de usuario

Como se ha señalado anteriormente, al haber tal diversidad de dispositivos y la correspondiente movilidad, las interfaces de usuario cobran mayor importancia. La adaptabilidad será una característica clave para las interfaces de usuario. El hecho de que una aplicación se utilice con el ratón, con una pantalla táctil y en pantallas de diferente resolución deberá ser razón suficiente para tratar de separar la interfaz de usuario de la propia aplicación, y el sistema deberá proveer mecanismos para que de una forma genérica la interfaz se pueda adaptar a cada dispositivo. Habrá que mantener coherentemente el estilo de la interfaz, sea una interfaz vía voz, visual o de cualquier otro tipo, para no desorientar al usuario. En todo caso resulta interesante impulsar las interfaces multimodales sobre cualquier plataforma siempre que sea posible. En este sentido, tal y como se verá más adelante en los ejemplos desarrollados, la funcionalidad relacionada con la interfaz se ha mantenido aislada del resto de la lógica del programa

4.2 Middleware para la Adaptabilidad de Aplicaciones en Ejecución

Como resultado de las necesidades que se han detectado en el ámbito de aplicaciones heterogéneas sensibles al contexto, se propone el desarrollo de una nueva funcionalidad para el middleware [Uribarren+06a], que responda a las necesidades de movilidad y adaptabilidad de los componentes de las aplicaciones ubicuas. No sólo se pretende adecuar la interfaz de una aplicación entre los diferentes dispositivos que un usuario pueda utilizar, sino que se necesitan mecanismos para garantizar la funcionalidad ofrecida por cada aplicación. Para ello se ha detectado la necesidad de dotar al middleware con un mecanismo para el intercambio de componentes de las aplicaciones entre los diferentes dispositivos que cualquier persona pueda utilizar.

Otro requisito básico destacable es la necesidad de desarrollar e implementar mecanismos independientes entre plataformas (.NET, JVM, .NET CF, Java ME...) garantizando la comunicación interplataforma entre dispositivos heterogéneos asegurando así la interoperabilidad. Entre estos mecanismos se encuentran, la

utilización de métodos de descubrimiento comunes y mecanismos de plug & play software.

4.2.1 Kernel Runner (KR)

El Kernel Runner (KR) es el elemento básico del nuevo mecanismo del *middleware* y provee de los mecanismos necesarios para la comunicación, envío, carga y ejecución de paquetes de aplicaciones entre dispositivos. Cabe señalar que el KR es necesariamente dependiente de la plataforma en la que se va a ejecutar. Esto es, el KR no puede ser desarrollado utilizando tecnología.NET (Java) para ser ejecutado indistintamente sobre un sistema operativo de la familia Linux (Windows). En este sentido, dependiendo de la plataforma destino, los paquetes de las aplicaciones a utilizar también serán dependientes de la plataforma, por lo que será necesaria la utilización de la versión correcta de cada aplicación en cada sistema operativo.

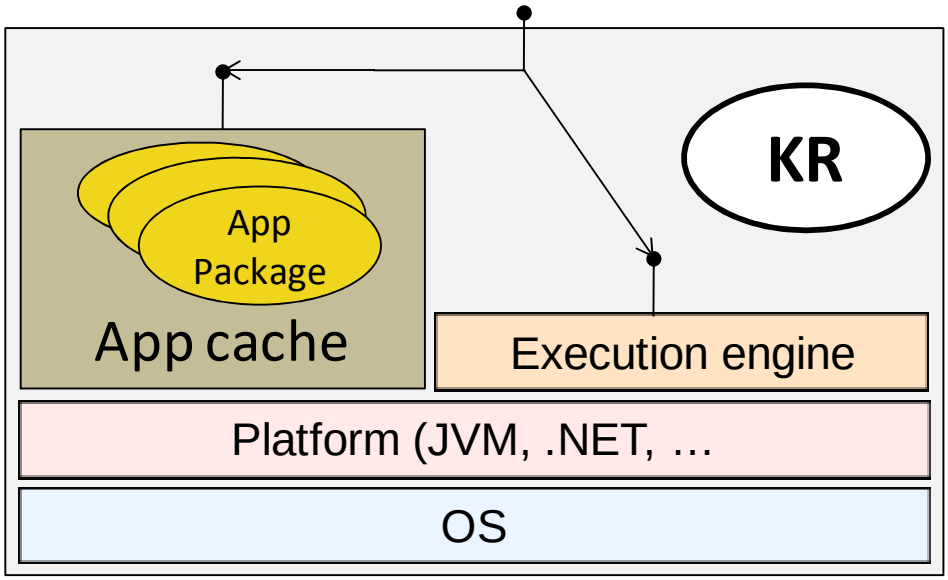


Fig. 4.1 Kernel Runner

El KR está compuesto por tres elementos: una interfaz, la caché (local) de aplicaciones y el motor de ejecución, tal y como se muestra en la figura 4.1.

La interfaz es la puerta de comunicación del KR por el que otros KR y aplicaciones se van a intercambiar información. En nuestra implementación, aunque se explicará con más detalle a posteriori, se ha utilizado el protocolo de interoperabilidad UPnP. Entre las características de UPnP se encuentra la capacidad de estandarización del descubrimiento de servicios y dispositivos, facilitando todo el mecanismo de autodescubrimiento y compartición de recursos en una red heterogénea en aras a conseguir los objetivos establecidos.

La caché de aplicaciones es un repositorio de aplicaciones en el que se encuentran las aplicaciones que pueden ser ejecutadas. Una aplicación externa podría chequear si se encuentra cierta aplicación disponible en la caché para poder ejecutarla utilizando el método ‘ExistApp’. Si no se encuentra en ella, se descarga la aplicación utilizando

'UploadApp' desde el repositorio correspondiente y se almacena localmente para poder ejecutarla posteriormente. Si al contrario, la versión de la aplicación correspondiente existiese, no haría falta descargarla o moverla de ningún sitio, ya que todo estaría listo para poder ejecutarse.

El motor de ejecución se encarga de ejecutar los programas que se encuentran en la caché de aplicaciones. Es capaz de ejecutar ciertas acciones asociadas a un programa tales como 'StartExecution' para lanzar la ejecución, 'StopExecution' para parar su ejecución, 'CheckRunning' para chequear si está en ejecución o 'OnlineReloadParameters' para actualizar dinámicamente ciertos parámetros de configuración del programa. Se utiliza el mecanismo de 'reflection' de las plataformas tanto .NET [ReflectionNET10] como Java [ReflectionJAVA10] para poder cargar componentes y en tiempo real ejecutar los programas en caliente.

4.2.2 Paquete de aplicación (AP)

Para facilitar la distribución y ejecución de las aplicaciones entre los distintos dispositivos provistos de KR, se ha creado lo que se ha denominado Paquete de Aplicación (AP, *Application Package*) por cada programa o aplicación. Un AP no es más que un fichero XML que almacena dos tipos de información: por un lado, los parámetros de configuración de la aplicación y, por otro lado, todos los ficheros (ejecutables y/o DLLs, en formato Base64) necesarios para la ejecución de la aplicación. Tal y como se ha mencionado anteriormente, utilizando los mecanismos de 'reflection' se ejecutarán las aplicaciones sin que anteriormente hubiera que instalarlos como en la forma clásica. Dentro del AP se incluyen los archivos (extensiones .EXE y .DLL) necesarios para la ejecución. Dichos ficheros ejecutables se almacenan serializados como cualquier otro parámetro dentro del fichero XML, y servirán al motor de ejecución (execution engine) para lanzar los programas.

Con el uso de los paquetes de aplicación, se consiguen separar los ficheros de ejecución de los parámetros de ejecución, pudiendo reutilizar y reasignar los parámetros de ejecución en diferentes paquetes para una misma aplicación.

Por otro lado, el uso de ficheros XML no es casual. El formato de fichero XML es soportado por todas la plataformas, siendo otro elemento importante a la hora de la interoperabilidad entre dispositivos al igual que el anteriormente mencionado UPnP. El contenido de los ficheros XML puede ser reutilizado por los diferentes dispositivos que están siendo ejecutados sobre cualquier sistema operativo y/o plataforma y/o lenguaje de programación. En la figura 4.2 se muestra por ejemplo el aspecto del contenido de un AP en XML.

Como se puede apreciar, aparte de información genérica como el nombre de la aplicación, versión,… también están incluidos los nombres de los ficheros DLL y el código de los mismos en Base64 'PackCode' y 'Code' que servirá para su posterior ejecución. También se indica cual es la función inicial al que el programa ha de llamar para comenzar la ejecución de la misma 'InitMethod' y los parámetros con los que

debe iniciarlo 'InitParams'. Asimismo, se empaquetan los ficheros de las DLLs cruzadas que son necesarias para la ejecución.

```
<Application xmlns:app="urn:application">
  <app:ApplicationName>MediaPlayer</app:ApplicationName>
  <app:Version>1.2</app:Version>
  <app:Description>Reproductor MediaPlayer</app:Description>
  <app:Platform>.NET Framework</app:Platform>
  <AssemblyName>EjemploAplicacionLanzadaPorReflection.dll</AssemblyName>
  <app:AssemblyFullName>EjemploAplicacionLanzadaPorReflection,
    Version=1.0.1740.195,Culture=neutral,
    PublicKeyToken=null</app:AssemblyFullName>
  <app:AssemblyCode>TVqQAAMAAAAEAAAA//8AALgAAAAA...</app:AssemblyCode>
  <app:InitForm>frmInicio</app:InitForm>
  <app:InitMethod>Inicializar</app:InitMethod>
  <app:InitParams>
    <app:Param>
      <Name>URL_BS</Name>
      <Type>string</Type>
      <Value>172.16.6.4:22000</Value>
    </app:Param>
    <app:Param>
      <Name>Idioma</Name>
      <Type>string</Type>
      <Value>es-es</Value>
    </app:Param>
  </app:InitParams>
  <app:EndMethod>Finalizar</app:EndMethod>
  <app:EndParams />
  <app:ReferencedAssemblies>
    <app:ReferencedAssembly>
      <AssemblyName>SHDoc.dll</AssemblyName>
      <AssemblyFullName>SHDoc, Version=1.0.1740.20316,
      Culture=neutral, PublicKeyToken=null</AssemblyFullName>
      <Code>TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAA...</Code>
    </app:ReferencedAssembly>
  </app:ReferencedAssemblies>
</Application>
```

Fig. 4.2 Formato XML del paquete de aplicación

4.2.3 Dispositivo Controlador descubrible (DDC)

Se define como Dispositivo Controlador Descubrible (DDC, *Discoverable Device Controller)* todo componente capaz de descubrir servicios en la red y utilizarlos. Un ejemplo de un DDC puede ser un 'control point de UPnP' que puede descubrir dispositivos UPnP y acceder a los servicios ofrecidos por los mismos. En este sentido, toda aplicación que quiera hacer uso de la plataforma middleware y más concretamente de la funcionalidad de mover aplicaciones y configuraciones, deberá implementar o poseer esta característica.

4.2.4 Otros Componentes

Se han identificado, aparte del KR, otra serie de elementos y funcionalidades que son necesarias para el *middleware*. Esta serie de nuevas funcionalidades o componentes sería común a cualquier tipo de desarrollo o escenario que queramos realizar y que utilicen los mecanismos fundamentales anteriormente descritos.

4.2.4.1 Repositorio de aplicaciones (AR)

El Repositorio de Aplicaciones (AR, Applications Repository) como su nombre indica, es un repositorio donde se encuentran los paquetes de aplicaciones disponibles para su descarga por cualquier aplicación. Su ubicación puede ser cualquiera, y en una misma red pueden coexistir varios AR. Tal y como se ha descrito anteriormente, por cada aplicación se genera un fichero XML, (Fig. 4.3) con toda la información necesaria para su ejecución. Los ficheros XML de aplicaciones se almacenan en el AR, proporcionando una descripción general de cada una de las aplicaciones disponibles.

```
<ApplicationsList xmlns:app="urn:ApplicationsList">
  <app:Applications>
    <app:Application>
      <Nombre>MediaPlayer</Nombre>
      <Plataforma>.NET Framework</Plataforma>
      <Version>1.2</Version>
      <Descripcion>Reproductor MediaPlayer</Descripcion>
    </app:Application>
    <app:Application>
      <Nombre>MediaPlayer</Nombre>
      <Plataforma>Linux</Plataforma>
      <Version>1.0</Version>
      <Descripcion>Reproductor MediaPlayer</Descripcion>
    </app:Application>
    <app:Application>
      <Nombre>MediaPlayer</Nombre>
      <Plataforma>Java</Plataforma>
      <Version>1.5</Version>
      <Descripcion>Reproductor MediaPlayer</Descripcion>
    </app:Application>
  </app:Applications>
</ApplicationsList>
```

Fig. 4.3 Formato XML del fichero del repositorio de aplicaciones

Las funciones básicas que ofrece el AR, a través de servicios web (genéricas), son: 'SearchApp' para conocer si se encuentra disponible cierta aplicación, 'DownloadApp' para obtener cierta aplicación, y 'UploadApp' para guardar cierta aplicación.

4.2.4.2 Repositorio de configuraciones (CR)

Al igual que el AR, el Repositorio de Configuraciones (CR, *Configurations Repository*) almacena la información de la parte correspondiente a las configuraciones en formato XML. En este sentido se ha visto la necesidad de guardar la configuración de ejecución de cada aplicación por nodo o dispositivo de ejecución y por usuario. De esta manera se dispone de la configuración personalizada y de las preferencias de cada usuario por ubicación.

Tal y como se puede apreciar en la figura 4.4, por un lado, podemos guardar la última configuración y preferencias del usuario y, por otro, esto nos ayuda a realizar diferentes trazabilidades a nivel de uso, estadísticas, históricos… En cada escenario se puede personalizar el criterio de almacenamiento de la información.

```
<ProfileList xmlns:app="urn:ProfileList">
  <app:ProfileList>
    <app:Profile>
      <Ubicacion>Sala Reuniones</Ubicacion>
      <Nodo>Home Cinema</Nodo>
      <Usuario>Auribarren</Usuario>
      <Aplicacion>Reproductor musica</Aplicacion>
      <Volumen>Auribarren</Volumen>
      <Canal>Radio Vitoria</Canal>
    </app:Profile>
    <app:Profile>
      <Ubicacion>Sala Reuniones</Ubicacion>
      <Nodo>Mesa</Nodo>
      <Usuario>Anonimo</Usuario>
      <Aplicacion>Control luces</Aplicacion>
      <Nivel>80</Nivel>
    </app:Profile>
  </app:ProfileList>
</ProfileList>
```

Fig. 4.4 Vista fichero configuración en el CR

4.2.4.3 Repositorio de recursos (RR)

Del mismo modo que AR y CR, el Repositorio de Recursos (RR, Resources Repository) almacena en formato XML la ubicación de repositorios externos de música, fotos, vídeos, entre otros, que en un momento dado un usuario puede necesitar y que de manera dinámica puede descargar o visualizar.

4.2.4.4 Sensores de contexto (CS)

Los Sensores de Contexto (CS, Context Sensor) son los elementos, componentes o dispositivos que informan del estado del entorno o contexto y de cualquier tipo de cambio: la información relativa a localización, temperatura, alarmas, estado de ventanas y/o de puertas, detectores de humo y/o fugas de agua, etc. Cualquier tipo de cambio es notificado (mediante eventos UPnP en el ejemplo desarrollado) a los

componentes subscritos a dichos dispositivos. Esta información es vital para detectar la movilidad y los cambios en el entorno y actuar en consecuencia, reconfigurando el comportamiento del sistema y de las aplicaciones involucradas.

4.2.4.5 Gestor de configuración y movilidad (MCM)

El componente Gestor de Configuración y Movilidad (MCM, Mobility Configuration Manager) es el responsable del descubrimiento y control de los diferentes KR en la red. Tiene la capacidad de mover, ejecutar y parar las aplicaciones en los diferentes KR. Este componente puede ser independiente o estar incluido dentro de otro middleware o programa para cumplir con los objetivos marcados. El MCM es el controlador necesario para llevar a cabo el análisis y la posterior movilidad de las aplicaciones pertinentes para obtener los resultados prefijados. Su función es la de orquestar un sistema. La orquestación en este caso es la habilidad de un sistema o componente de sincronizar las acciones y/o los eventos entre el resto de sistemas con los que se interrelaciona.

En función de la lógica predefinida o programada, el MCM chequea si en cierta ubicación con KR existe un paquete de aplicación; si no existiese, localizaría un AR desde donde descargar el paquete de la aplicación y lo mueve al KR correspondiente. Acto seguido mueve el fichero de configuraciones para ese paquete y ya puede lanzar la ejecución del paquete remotamente. Será responsabilidad del MCM el coordinar y orquestar los diferentes KR y los repositorios involucrados.

4.2.4.6 Gestor de contexto (CM)

La adaptación dinámica de aplicaciones requiere de la capacidad de actuación en respuesta a los cambios de contexto en un entorno. Estos cambios son detectados por los CS. Cuando uno o varios CS notifican un cambio, el Gestor de Contexto (CM, Context Manager) utiliza esta información para evaluar y generar la respuesta pertinente que podrá ser la entrada necesaria para cualquier MCM para que pueda realizar los movimientos necesarios de los programas y paquetes involucrados.

La inteligencia que a este tipo de sistemas se le atribuye no es más que una sentencia tipo ‘Si se cumple esto, entonces haz esto’, lo que en la literatura se conoce como ‘ECA rules’ (Event-Condition-Action). Existen también sistemas de deducción más complejos basados en probabilidades o en técnicas de Inteligencia Artificial. Estas ECA rules y sistemas basados en reglas se estudiarán en el siguiente capítulo. En cada caso de implementación, dependiendo de los requisitos a cumplir, el CM podrá tener un motor de inferencia más o menos complejo.

El CM no se considera un componente imprescindible en la arquitectura propuesta, como pueden ser el KR o los repositorios, ya que el CM puede llegar a ser personalizado y en algunos casos, la función del CM puede llegar a estar integrada como parte de componentes de otros middlewares o aplicaciones. De esta manera, la evaluación del contexto y generación de las órdenes de comportamiento pueden llegar

a ser generadas por sistemas externos, y que serán utilizadas por el MCM para mover y recrear el comportamiento de los entornos deseados.

4.2.5 Varias consideraciones

En general, los entornos ubicuos son no deterministas respecto al número de usuarios que forman parte de ella, disponibilidad de recursos y dispositivos conectados. Habrá elementos que se mantienen estables en el tiempo y otros que cambien. Para que el usuario perciba de manera transparente los cambios del entorno, será imprescindible que el sistema reaccione ante los cambios de manera imperceptible.

Por otro lado, debido a la movilidad del usuario y a las restricciones de capacidad de procesamiento simultáneo y almacenamiento de ciertos dispositivos móviles, no es posible que un usuario se lleve consigo todos sus programas preferidos y ejecutarlos. Es por ello que para poder responder a las necesidades de acceso a todo tipo de información de forma transparente y ágil haya que proveer de otros mecanismos. La replicación y la posibilidad de disponer de sistemas y fuentes de información disponibles online 24 horas será necesaria. Para ello se utilizan fuentes de información o repositorios estáticos accesibles en cualquier momento para poder dar la respuesta requerida al usuario en cada momento. La arquitectura propuesta consta de nodos y ubicaciones estáticas que sirven datos para recrear los entornos dinámicos requeridos por el usuario.

Para finalizar con la descripción del *middleware*, remarcar que los componentes anteriormente descritos son en sí mismos partes que cualquier sistema operativo genérico podría tener. Pero llegados a este punto, cabe destacar el nivel de interoperabilidad obtenido, se demuestra que se pueden mover aplicaciones y sus configuraciones entre dispositivos que utilizan sistemas operativos y programas de desarrollo basados en diferentes plataformas, intentando mantener las funcionalidades de las aplicaciones. A continuación, en el apartado 4.3.3, se describe y valida con un ejemplo cómo funciona la movilidad de aplicaciones.

4.3 Validación del sistema

Como ejemplo para mostrar la validez del *middleware* y así lograr la movilidad y configuración de aplicaciones/servicios con respecto a la ubicación del usuario, a continuación se va a describir el escenario en el que se han probado e implementado los componentes descritos anteriormente.

4.3.1 Descripción del escenario

El tipo de escenario elegido ha sido del tipo ‘Follow me’ [Satyanarayanan04]. En este escenario, en función de los cambios de localización de un usuario, el sistema de manera automática mueve las aplicaciones que se están ejecutando en los dispositivos involucrados. Por otro lado, mediante el uso de mecanismos descritos anteriormente, tales como la obtención y guardado de las configuraciones descritas en

el apartado 4.2.4 adicionalmente se ajustan otros parámetros del entorno, tales como la luz ambiental, nivel de música, etc. a las necesidades y preferencias del usuario.

Por otro lado, para que el conjunto del sistema cumpla con los requisitos de interoperabilidad descritos en la introducción de este capítulo, se han utilizado los principios que se han descrito en el capítulo 3 de esta tesis. En concreto, en el ejemplo que se describe a continuación se ha utilizado UPnP como protocolo de interoperabilidad.

4.3.2 Descripción de los elementos del entorno

En cuanto a la infraestructura hardware necesaria, el lugar de trabajo se ha dotado de diferentes dispositivos: lectores y tarjetas RFID para localización e identificación de usuarios. Los lectores RFID (sensores de contexto) de localización de usuarios han sido ocultados debajo de las mesas, de manera que cuando llegamos a una ubicación (mesa) y dejamos nuestro bolso o cartera encima de la mesa (incluyendo una tarjeta RFID), la tarjeta RFID es leída y el CM y MCM gestionan dicha información. También se han utilizado dispositivos de encendido y apagado de lámparas conectando las líneas serie de los PCs involucrados con una interfaz de UPnP. Por otro lado, en la red también se han configurado y habilitado en varios PCs los diferentes repositorios AR, CR y RR. Para finalizar, se han generado varios paquetes de aplicaciones como son un navegador de internet y un reproductor de música utilizando tecnología .NET y Java. En la figura 4.5 se muestra una imagen donde aparecen los elementos citados anteriormente.

Fig. 4.5 Dispositivos utilizados

Para demostrar la validez del sistema propuesto, se han utilizado tres PCs localizados en una oficina. Uno de ellos se ha situado en la entrada de la oficina (PC1). En el PC1 se ha colocado uno de los lectores RFID, con el que detectaremos cuándo una

persona entra y sale de la oficina. Las aplicaciones del PC1 se ejecutan en Java sobre Linux. Los otros dos PCs llamados PC2 y PC3 estarán situados dentro de la oficina y se utilizarán para demostrar el movimiento de aplicaciones cuando un usuario cambia de ubicación. En este caso el sistema operativo de ambos será Windows XP y las aplicaciones se ejecutan sobre el .NET *Framework*. Para la ejecución, en los tres PCs se lanza el KR correspondiente, y el PC1 tiene el roll de RR en el que se encuentran disponibles música y un servidor de media. En la figura 4.6 se muestran los diferentes componentes instalados en los PCs de prueba.

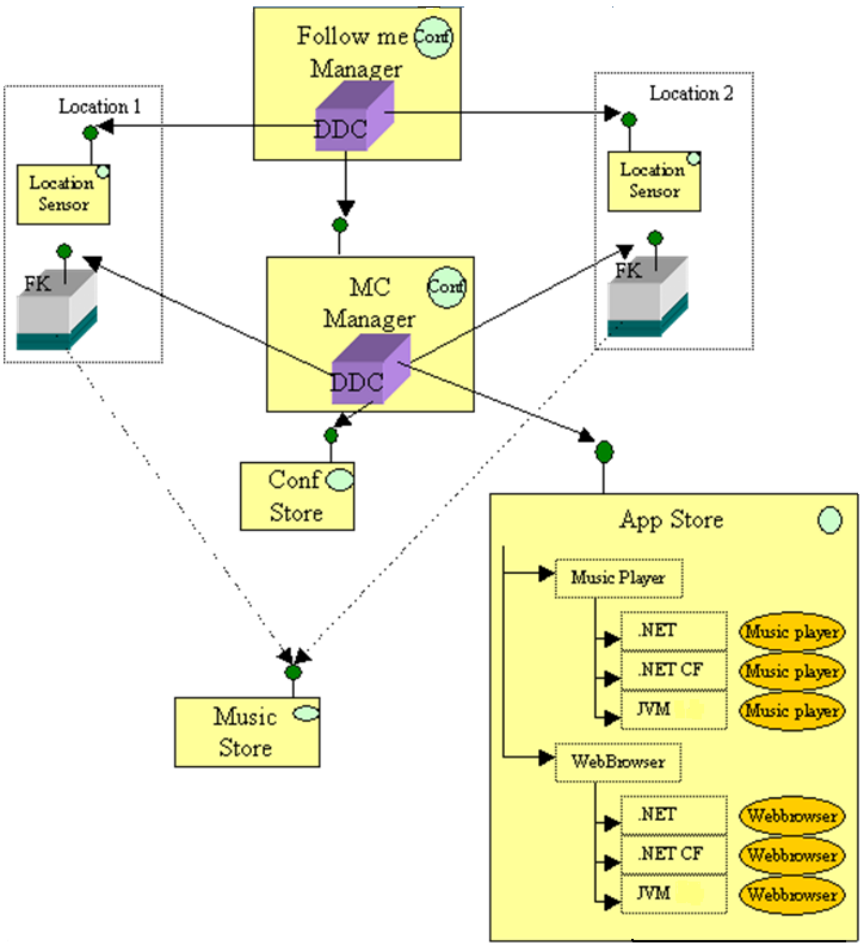


Fig. 4.6 Componentes escenario 'Follow Me'

Para terminar, señalar que en la realización de la plataforma de demostración, el KR, la aplicación del navegador y los CS se han desarrollado en VB .NET y en Java 1.6, probándolos sobre MS Windows y sobre Linux; por otro lado el CM, MCM, RR, CR y AR se han desarrollado en VB .NET. El protocolo de interoperabilidad utilizado en el *middleware* ha sido UPnP y los detectores de presencia RFID se conectaron a la línea serie y USB de los PCs en los que se ejecutó el KR y sobre él la aplicación de control.

4.3.3 Ejemplo

A continuación, mediante un ejemplo se mostrará cómo funciona la movilidad de aplicaciones. Juan trabaja en una oficina y el PC que utiliza habitualmente es el PC2,

aunque el PC1 se encuentra en la entrada hacia su puesto de trabajo. Juan llega a la oficina y pasa por la puerta, el detector RFID del PC1 situado en la entrada de la oficina notifica al CM de dicho evento. El CM analiza que Juan está entrando en la habitación. A su vez, el CM analiza qué acción debe realizar cuando Juan entra en la oficina. Se ha establecido que a Juan le gusta la música clásica. Por lo tanto, el CM envía un mensaje al MCM para que lance la aplicación de música con la opción de música clásica en el PC2 de Juan. El MCM, antes de poder ejecutar nada, chequea preguntando al KR del PC2 si existe la aplicación reproductora de música disponible, y si no es así la descarga del AR del PC1, y a continuación lo lanza con la música clásica en este caso. Para cuando Juan llega a su puesto de trabajo PC2, la música ya está sonando. En la figura 4.7 se muestra el diagrama de secuencia en el que se muestra el orden de ocurrencia de los eventos.

En cada PC, se ejecuta una aplicación general por defecto, desarrollada para ayudar en el proceso de depuración y optimización de diferentes aspectos que muestra los paquetes disponibles y/o instalados, parecido a la opción de 'Agregar/Quitar programas' del panel de control del sistema operativo MS Windows. Utilizando esta aplicación, al usuario se le muestran todos los programas disponibles en los diferentes repositorios. Si se elige un programa de esa lista, por ejemplo, Juan selecciona el navegador de internet, esta aplicación general comunica al MCM que Juan quiere ejecutar el navegador y el MCM se encarga de chequear si está localmente disponible en el KR y, si no es así, lo descarga del AR y lo ejecuta sobre el KR local. Para Juan es como si el navegador web estuviera instalado localmente. Juan ya puede leer el periódico online preferido.

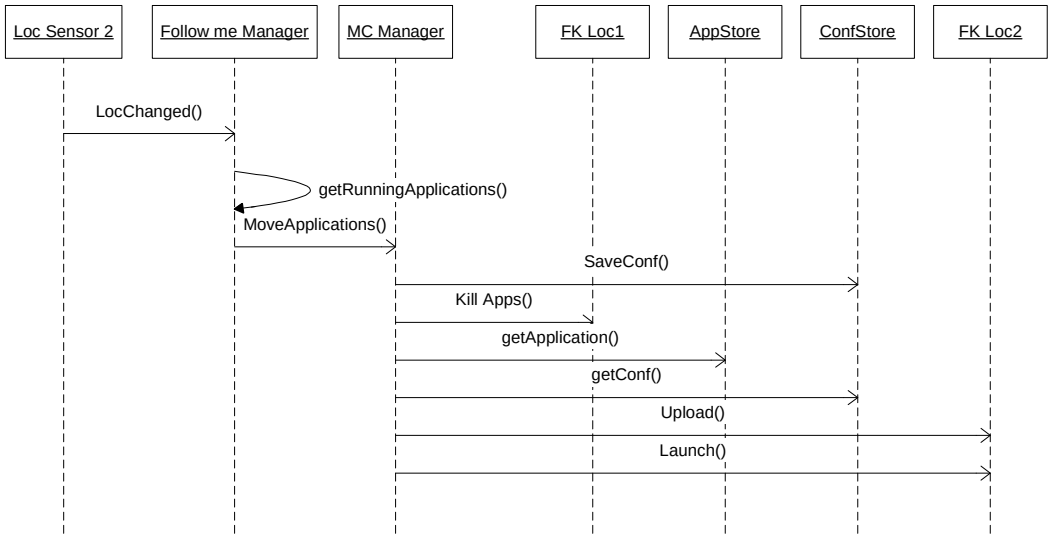


Fig. 4.7 Diagrama de secuencia de instalación de una aplicación

Juan decide moverse al PC3. El sistema RFID detecta una nueva lectura. El CS del RFID envía un mensaje CM de que Juan se encuentra en una nueva ubicación. El CM chequea que Juan se encontraba trabajando en el PC2 y que ha dejado aplicaciones

en marcha. Por lo tanto, decide mover dichas aplicaciones al PC3. El CM primero guarda en el CR las opciones de configuración de los programas en ejecución, tales como el último link del navegador en el que estaba Juan. A continuación ordena al MCM para que instale y ejecute los paquetes en el PC3 con la última configuración. El navegador es ejecutado con el último link conocido.

4.4 Conclusiones

Se ha estudiado, analizado, diseñado y desarrollado un *middleware* para permitir el despliegue y la ejecución de aplicaciones en un entorno ubicuo respondiendo a los requisitos de movilidad, y más concretamente a los cambios de ubicación de los usuarios. Tal *middleware* ofrece las siguientes dos capacidades para:

Mover aplicaciones independientemente de la plataforma de ejecución entre distintos dispositivos.

Mantener y/o adaptar las preferencias y configuraciones de ejecución entre las diferentes ubicaciones.

Gracias a que los elementos se auto-descubren basándose en los fundamentos (heterogeneidad e interoperabilidad) analizados en el capítulo 3, la conexión y el descubrimiento de componentes software y hardware es transparente para el usuario.

Así pues, este *middleware* ofrece el soporte para crear aplicaciones portables, reconfigurables, y cuya ejecución pueda ser sobre distintos dispositivos y plataformas heterogéneas. Esta función o mecanismo del *middleware*, junto con lo que se explicará en el siguiente capítulo de esta tesis, dotarán al *middleware framework* propuesto por esta tesis de la capacidad para realmente adaptar y reconfigurar el comportamiento de prácticamente cualquier contexto o entorno ubicuo que se pueda plantear en un entorno, como el hogar u oficina, a los deseos y necesidades de los usuarios.

No obstante, las aplicaciones sensibles al contexto no solo han de contemplar la movilidad de aplicaciones relacionada con la ubicación del usuario. Tal y como se verá en el siguiente capítulo, dicha personalización podrá ser llevada a cabo mediante mecanismos de configuración de entornos automatizables siguiendo las preferencias establecidas por el propio usuario.

Capítulo 5: Configuración de entornos

Sobre gustos y colores no hay nada escrito, cada uno de nosotros tenemos gustos diferentes en el color, diseño y tamaño, entre otros. En nuestra casa, elegimos el color de las paredes, el diseño de los muebles y el tamaño de la televisión. Al principio, la mayoría de los dispositivos de nuestro hogar eran estáticos, sin opción de configurarlos; sin embargo, hoy en día, por ejemplo en dispositivos como la televisión, podemos cambiar el contraste, el brillo y controlar que se apague al cabo de un tiempo determinado. De la misma forma, sería deseable que pudiéramos configurar y personalizar nuestro entorno o ambiente (por ejemplo, nuestro hogar, la oficina, un almacén, un hospital) de acuerdo con nuestros gustos, para que se encienda automáticamente la alarma de seguridad cuando no haya nadie en casa o por ejemplo, se encienda la cafetera cuando me levante de la cama. Opciones que permitirán aumentar el confort, el ahorro energético, la seguridad y disfrutar de nuevas formas de ocio y entretenimiento. Este interfaz y su evaluación se estudiarán en el siguiente capítulo.

5.1 Introducción

Actualmente, la mayoría de los entornos inteligentes ofrecen unos determinados servicios al usuario, como el de encender las luces cuando alguien entra en una habitación, controlar el estado encendido y apagado de un electrodoméstico u otro dispositivo. Sin embargo, muchas veces queremos un entorno más complejo, que involucre no solo un único dispositivo sino un ecosistema variado de dispositivos, lo que supone un gran desafío. Aspectos interesantes a considerar son: a) cuando el control involucra coordinar varios dispositivos simultáneamente, no hay modo de hacerlo a no ser que se definan unas reglas de comportamiento y b) el problema es que dichas reglas de comportamiento deberían tener en consideración los usuarios actualmente presentes en el entorno. Esto último implica la consideración de un interfaz o editor de reglas de fácil manejo, multiusuario y la provisión de mecanismos para la resolución de conflictos entre varios usuarios. Desde un punto de vista de programador y de usuario final, esto es realmente retador. Por ejemplo, la acción de “Encender la alarma cuando no hay nadie en casa” involucra más de un dispositivo y podría ser una opción de configuración interesante para el usuario. Iniciativas como Pronto de Philips, un dispositivo con una interfaz para controlar dispositivos de audio y video, contenido multimedia y luz, están introduciendo en el mercado la idea de tener un único interfaz que controle la configuración y personalización de los entornos [Pronto09]. Todos los aspectos relativos a la configuración y personalización por parte

de los usuarios se estudiarán en el capítulo siguiente, mientras que en este capítulo se estudiará cómo llevar a cabo la configuración de un entorno.

Un claro ejemplo es un edificio equipado con sensores de movimiento que pueden detectar si una persona está en una determinada zona y encender la luz. En este caso, los sensores de movimiento harán que se enciendan las luces cuando se detecte movimiento y las apagarán automáticamente cuando no detecten movimiento durante un periodo de tiempo. Estos sensores de movimiento realizan una tarea específica, que es encender o apagar la luz (un modo único de funcionamiento) y su comportamiento/actuación se ofrece directamente al usuario sin configuración. Sin embargo, por una parte, los nuevos avances en sistemas embarcados, micro controladores y redes de sensores inalámbricos (por ejemplo, motes) están jugando un rol importante en cambiar ese modo único de funcionamiento y ofrecer un nuevo control por parte del usuario.

Las características de nuestros hogares, sitios de trabajo e infraestructuras están cambiando con las nuevas tecnologías de la información y del conocimiento y se están empezando a introducir nuevos elementos que proveen servicios para mejorar dichos entornos para, por una parte, mejorar las capacidades del usuario y por otra su calidad de vida. Sin embargo, la complejidad de crear comportamientos ambientales personalizados, o en general servicios personalizados, está creciendo muy rápido con la llegada y el uso de una gran variedad de dispositivos (como son los dispositivos embarcados) junto con una funcionalidad cada vez más compleja y no un único modo de funcionamiento. Este gran número de funcionalidades embarcadas y las combinaciones de éstas ofrecen al usuario un sin fin de nuevos servicios en su entorno.

Este trabajo se sitúa dentro de la computación sensible al contexto o ‘context-aware computing’. Los sistemas context-aware hacen uso de información situacional o contexto para dar información relevante y nuevos servicios a los usuarios. En estos sistemas, existe una red de sensores, actuadores y controladores que dan información de contexto del entorno, y los dispositivos son controlados automáticamente según la información de contexto provista por estos dispositivos. La información de contexto puede ser: la posición del usuario, la hora, la temperatura de una habitación, que la lavadora ha terminado, etc. En un primer paso, es necesario identificar el contexto actual del entorno, es decir, descubrir la red de dispositivos disponibles en el entorno y en segundo lugar, es necesario que dichos dispositivos se comuniquen y colaboren entre ellos. En este capítulo, se usarán las soluciones adoptadas en estos campos y descritas en los Capítulos 3 y 4.

En esta tesis, se distinguen los términos configuración y personalización. Se considera que el primer termino se refiere a qué se percibe en el entorno y a cómo se actúa según los cambios producidos por la inclusión o desaparición de dispositivos, realizando una configuración inicial de los dispositivos con parámetros predefinidos o por defecto, adaptándolos a los requisitos establecidos. En cambio, la personalización se refiere a cambiar el comportamiento o funcionalidad de un entorno en vivo, según

las necesidades y preferencias puntuales de los usuarios. En el capítulo 6 se abordará la personalización. En algunos casos, se ha utilizado el término configuración para referirse a ambos conceptos.

Los principales retos de investigación de este capítulo son:

- Crear un motor de reglas que permita gestionar todo el conjunto de reglas que define el comportamiento de un entorno, para que:
    i. Las reglas se evalúen y ejecuten de una manera eficiente.
    ii. Las reglas se adapten/configuren automáticamente ante los cambios de contexto.
    iii. Se soporten reglas temporales, ejecutadas en cierto periodo de tiempo.
    iv. Se detecten conflictos o duplicidad de reglas en el momento de la creación.
    v. Se detecten conflictos entre reglas de distintos usuarios y se resuelvan en base a las prioridades y roles asignados tanto a los usuarios como a las reglas en ejecución y en tiempo real.
    vi. Se puedan añadir o quitar reglas en tiempo de ejecución.

En el apartado siguiente, se introduce brevemente el paradigma ECA (Event-Condition-Action), que servirá para entender los conceptos básicos usados en los siguientes apartados. Este paradigma dará solución a la problemática de la ejecución de reglas para la configuración de un entorno, siendo ésta a priori suficiente para abordar la coordinación de los sistemas ubicuos.

5.2 Paradigma ECA

Para entender el paradigma ECA, se explicará primero la importancia de los eventos y, a continuación, se describirá qué es una regla ECA (denotada generalmente por regla) y las diferentes partes que la componen.

5.2.1 Eventos

Mientras que los sistemas sin eventos están basados en controles periódicos o activación por tiempo incrementando la carga de trabajo y de comunicación, los sistemas basados en el paso de eventos realizan una o varias acciones únicamente cuando uno o varios eventos se producen. Los sistemas basados en eventos ofrecen una alternativa muy prometedora cuando se usan sistemas con capacidades reducidas de computación y de comunicación. Últimamente, estos están siendo utilizados satisfactoriamente en aplicaciones que van desde sistemas de tiempo real hasta, sistemas de adquisición de datos y control.

Otra ventaja que suelen tener estos sistemas es que permiten capturar información de diferentes fuentes (dispositivos) y distribuirla en una manera definida a los posibles consumidores. Por consumidores o suscriptores de eventos entendemos que son

aquellos sistemas que consumen eventos o, en otras palabras, que se han suscrito a la escucha de esos eventos. Generalmente, el mecanismo usado para la notificación de eventos se llama publicación-subscripción de eventos (servicios de notificación *Publisher-Subscriber*), y se utiliza para el intercambio de mensajes entre diversas fuentes (como pueden ser aplicaciones, sensores, sistemas monitores y controladores). Entonces, el cambio de estado de un objeto/variable automáticamente notifica a todos los suscriptores.

Las aplicaciones sensibles al contexto deben reaccionar al entorno dependiendo de los cambios de contexto que se produzcan, generalmente, estos cambios son notificados mediante eventos. Estas aplicaciones funcionan de la siguiente manera: reaccionan ante eventos y ejecutan una acción o un grupo de acciones, esto es, responden al modelo ECA (Evento-Condición-Acción). Así pues, cuando un evento ocurre, si una cierta condición se satisface, entonces se ejecuta la acción. El concepto de ECA está inspirado en el concepto de Bases de Datos Activas [Dittrich+96, Paton+99]. En los años 80, se trató de dar solución al problema de monitorizar y controlar cambios en las bases de datos, evitando así la intervención o búsqueda por parte de un usuario. Por primera vez, se definieron términos como eventos, condiciones y acciones [Chakravarthy+08] que se explicarán a continuación.

5.2.2 Reglas ECA

El concepto de regla surgió al intentar dar solución a la gestión de los cambios en las bases de datos. En general, en las bases de datos estas reglas se llaman triggers o disparadores y están basadas en el paradigma ECA, es por ello que también se denominan reglas ECA. Una regla ECA es una regla compuesta de evento, condición y acción y su estructura general es ON <event> IF <condition> THEN <action>. Evento, condición y acción se refieren a un único elemento o a varios. Por ejemplo, en una base de datos activa, una regla definida en dicha base de datos ejecutaría la acción que define, sobre la instancia que dispara el evento, siempre que la condición se verifique.

Cabe señalar que la creación y la ejecución de reglas en la literatura se han abordado desde diferentes ámbitos. En el campo de la Inteligencia Artificial, varios sistemas expertos han basado sus conclusiones en el conocimiento expresado en reglas [Allemang06, Esch04, NASA99]. Las reglas permiten representar relaciones entre dos o más objetos e incluye generalmente dos partes: la premisa y la conclusión. El motor de inferencia o de reglas obtendrá las conclusiones a través de dichas reglas.

En nuestro caso, el sistema configurador del entorno basado en reglas funcionará de una manera similar a las bases de datos activas o sistemas expertos basados en reglas, y el motor de reglas será el responsable de interpretar dichas reglas y ejecutarlas. Al producirse un cambio en el contexto (un evento), el motor analizará las condiciones afectadas y qué acción (o grupo de acciones) debe desencadenar.

5.3 Gestor de eventos

El componente de Gestor de Eventos (GE) nos provee de un mecanismo para el acceso y manejo de los aspectos relativos a la comunicación de los eventos. Para entender la gestión de eventos se necesitan conocer los siguientes términos: condición eventada, enrolamiento de un evento y acción del evento.

1. Condición eventada: condición que se evalúa si se produce el evento asociado a dicha condición.
2. Enrolamiento de un evento (event enrollment): a quién hay que notificar de la ocurrencia de un evento.
3. Acción del evento: acción a realizar al ocurrir el evento.

Una condición eventada es una condición que forma parte de una regla y que se evalúa si y sólo si cierta variable perteneciente a dicha condición ha cambiado su valor. El cambio de valor de dicha variable es recibido a través de un evento. Veamos con un ejemplo qué quiere decir lo anterior. Imaginemos que tenemos una regla del tipo “Si la temperatura es mayor que cierto valor, entonces enciende el aire acondicionado”, el motor de reglas hará lo siguiente:

- Primero, creará una variable lógica ‘temperatura’, después asociará dicha variable lógica con la variable temperatura del dispositivo real para poder recibir los cambios de valor. En otras palabras, las variables lógicas se suscribirán a las variables de estado de los dispositivos y estas formarán parte de las condiciones, siendo entonces condiciones eventadas. En nuestro caso se va a utilizar el mecanismo de patrón publicador/subscriptor, al igual que se hizo en [Pietzuch+03]. La implementación se ha llevado a cabo con .NET.

En la figura 5.1 se puede ver la manera en la que una aplicación puede llegar a interrelacionarse con un sistema publicador/suscriptor, utilizando una vía directa (parte izquierda de la imagen) o utilizando un intermediario (parte derecha de la imagen).

- Cada vez que en el dispositivo termostato se modifique la variable temperatura, la variable lógica asociada en la condición será notificada a través de un evento (variable eventada).
- Si esta variable lógica formara parte de una o varias condiciones, únicamente se evaluarían estas. El resto de condiciones no se evaluarán.
- La evaluación de una condición si únicamente es verdadera, generará la ejecución de cierta acción o de varias acciones (acción del evento).
- Hay que tener en cuenta que la ejecución de una acción provocará el disparo de más eventos que serán recibidos por aquellos elementos suscritos.

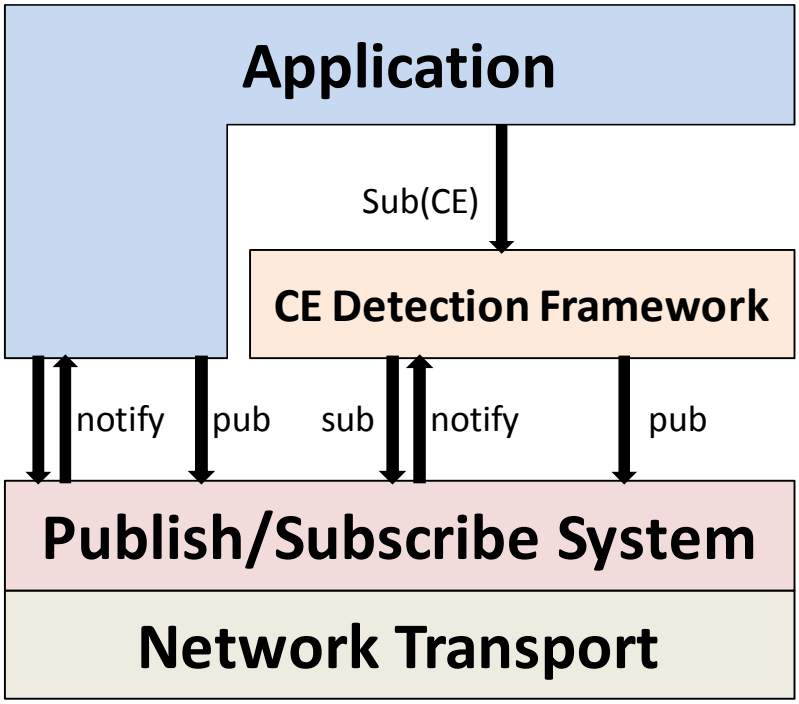


Fig. 5.1 Mecanismo publicador/subscriptor

Para mejorar la eficiencia del sistema y así lograr que los eventos sean gestionados de una forma paralela, por cada condición se creará un hilo de ejecución o *thread* independiente, es decir, un hilo independiente que se encargará de su propia gestión de eventos y evaluación de condiciones. Esto es posible gracias al enrolamiento del evento, esto es, al asociar directamente la recepción de un evento que es enviado por la variable concreta del dispositivo con la variable virtual o lógica del sistema.

Las arquitecturas clásicas de los motores de reglas ECA centralizaban la entrada y la gestión de todos los eventos en un único componente que evaluaba la lista de condiciones una por una, incluso si la variable involucrada no estaba en la condición, intentando buscar correspondencias entre el evento que había llegado y las condiciones definidas. Claramente, se destaca la poca eficiencia de este mecanismo frente al usado en esta tesis.

El mecanismo de asociación utilizado en este trabajo es directo, ya que no necesita de intermediarios como en las aproximaciones al paradigma del patrón publicador/suscriptor [Eugster+03], o como en el patrón 'Whiteboard' [Whiteboard04] desarrollado para la plataforma OSGi donde un sistema centralizado se encarga de recibir los eventos y de repartirlos a quiénes así lo hayan solicitado. El tipo de comunicación es P2P (parte izquierda de la figura 5.1) frente al otro en el que interviene un servidor o nodo central que hace de intermediario (parte derecha de la figura 5.1), utilizando el llamado CE (*common event detection manager*). Es la propia variable lógica la que mantiene una lista de sus clientes a los que tiene que enviar la notificación solicitada o suscriptores. En este sentido, la gestión queda en manos de los clientes. Por ello, cuando ya no interese estar 'conectado', será responsabilidad del

cliente de-suscribirse de la variable, utilizando los mecanismos que el entorno de programación utilizado permita, terminando con la relación que los mantenía unidos.

5.4 Motor de reglas (MR)

Las arquitecturas clásicas de los motores de reglas funcionaban de la siguiente manera: (1) centralizaban la entrada y gestión de todos los eventos en un único componente, tal y como se ha descrito anteriormente o bien no usaban eventos, (2) a continuación, se recorría toda la lista de condiciones de cada una de las reglas, incluso si la variable involucrada no estaba en la condición, intentando buscar correspondencias entre el evento que había llegado y las condiciones definidas. Este proceso se repetía siempre que había un cambio en una de las variables.

En nuestro caso, por el contrario, el sistema se encuentra en estado de hibernación 'idle' hasta que recibe un evento, al contrario que los sistemas clásicos en los que hay un bucle infinito que se está ejecutando cada cierto tiempo evaluando y ejecutando las reglas en función de los valores actuales del sistema. A diferencia de los motores de reglas clásicos, los siguientes elementos: variable, condición y regla estarán a la espera de recibir eventos. Entonces se evaluará sólo la parte correspondiente al evento involucrado y generará otro evento o mensaje que será la entrada o punto de partida para otro elemento, y así hasta que la tarea encomendada sea terminada. Es decir, si una regla consta de varias condiciones anidadas como por ejemplo:

```
AND     (OR (Ventana=abierta) (Puerta=abierta))
        (AND (Abuela=enCasa) (Niño=enCasa))
```

El evento de detección de que la abuela está en casa solo va a implicar la evaluación de la condición en la que se encuentra y el resto de condiciones como el OR no son evaluadas ya que no han sufrido cambios y se conoce su estado anterior. El cambio de una condición perteneciente a otra condición, en este caso el AND de rango superior, implica la evaluación de dicha condición como es natural.

Al estar enlazados por el paso de eventos (mensajes), la ejecución de unos componentes depende de los otros, de manera que si en un momento dado no se cumplen las condiciones necesarias en un componente, la ejecución derivada de un cambio de variable real se interrumpe en ese punto, terminando con la secuencia de ejecución. En los siguientes apartados se explica con detalle el flujo de información desencadenado por un cambio de variable real y por los diferentes componentes por los que pasa.

5.4.1 Middleware Framework

El *middleware framework* desarrollado hasta ahora da soporte a diferentes aspectos relacionados con el descubrimiento, enlazado o *'binding'* entre componentes y manejo

de los eventos entre los diferentes componentes del sistema. Se encarga también de dar soporte de ciertas funcionalidades necesarias para dar robustez al sistema, como puede ser la gestión de desconexiones en las comunicaciones o dispositivos desaparecidos, dotando al sistema de mecanismos para gestionar este tipo de excepciones, como se mostrará más adelante.

5.4.1.1 Componentes del motor de reglas

En la figura 5.2 se muestran los componentes básicos del motor de reglas y el flujo de la información que sigue un evento de entrada, desencadenando a su vez nuevos eventos entre los diferentes componentes.

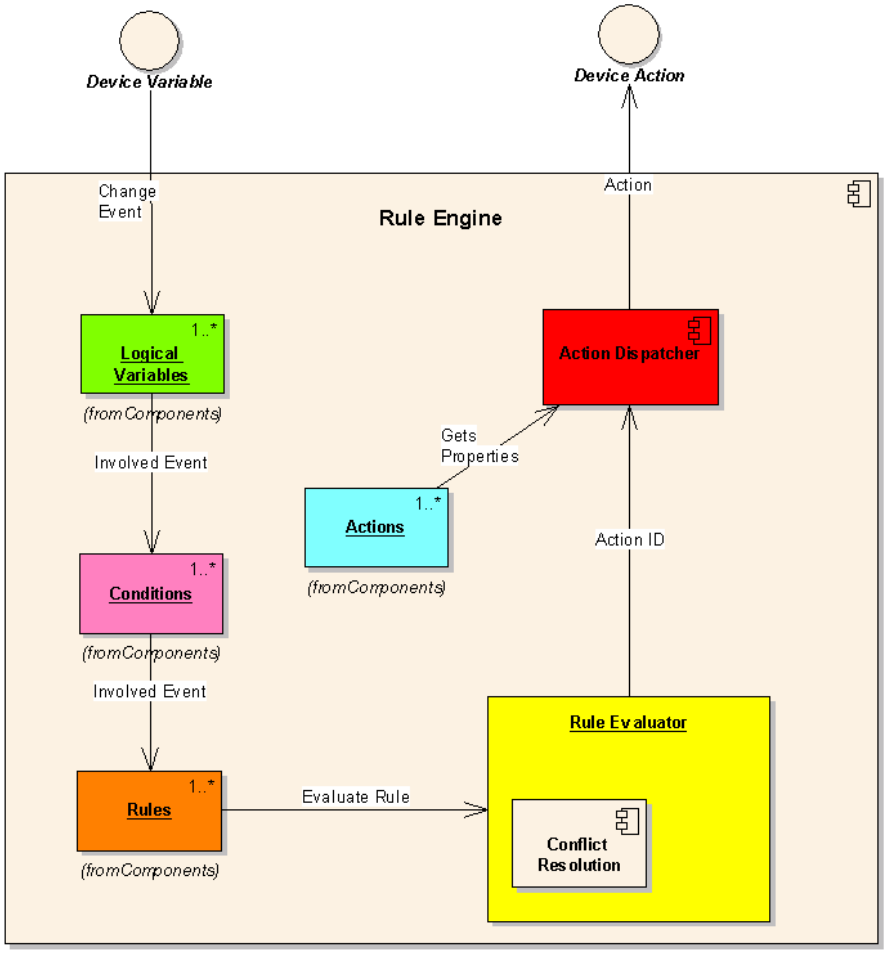


Fig. 5.2 Componentes del motor de reglas

- ✓ '*Device Variable*' se refiere a una variable del dispositivo, que puede ser por ejemplo, *On, Off o Status*, cuyo cambio produce un evento. Por cada variable del dispositivo, se crea una variable lógica en 'Logical Variables'. 'Logical Variables' engloba una lista de variables lógicas asociadas cada una de ellas a una única variable del dispositivo. Esto significa que la variable lógica se suscribe al evento producido por un cambio en el valor de la correspondiente variable asociada, es decir, cuando una variable de un dispositivo cambia su valor, esto provoca que se reciba un evento en la variable lógica suscrita. Al

producirse un evento, las variables lógicas suscritas a tal evento lanzan otro evento que provocará la evaluación de la condición o condiciones (indirectamente aquellas condiciones cuya variable eventada forma parte de), siguiendo el proceso explicado anteriormente en el apartado 5.4 para el paso de eventos dentro de la propia regla. Si un dispositivo no se ha anunciado o desaparece de la red, sus variables lógicas se mantienen en el motor de reglas aunque su valor es actualizado. En este último caso, las variables pasan a un estado 'pasivo' y, dependiendo de la configuración del editor de reglas por el usuario, pasan a tomar un valor por defecto o mantienen el último valor conocido. De esta manera, se asegura que una regla con varias condiciones involucradas y alguna de ellas asociadas con variables de dispositivos que no están presentes en un instante de tiempo se puedan seguir evaluando con el valor establecido. Así, evitamos que el motor de reglas no produzca estados indeseados ni evaluaciones incompletas cuando falta por descubrir un dispositivo, por un fallo temporal en la red, desconexiones, etcétera. Se trata de un mecanismo como el de una válvula eléctrica que en caso de falta de corriente su estado por defecto, por ejemplo, es la de cerrado.

Existen también variables de tipo 'Tiempo', creadas expresamente para que ciertas condi*ciones se evalú*en en un momento determinado. De esta manera se pueden reflejar acciones que queremos que ocurran en un instante de tiempo concreto o durante un periodo de tiempo. Su comportamiento es similar al de una variable de un dispositivo que al cumplirse el tiempo definido provocan un evento para que a continuación se evalúe la condición o condiciones involucradas.

- ✓ '*Conditions*' consta de la lista de las condiciones de cada una de las reglas. Las operaciones para realizar condiciones compuestas soportadas por el MR vienen dadas por las operaciones relacionales =, <>, <, >, <= y >=, las operaciones matemáticas de suma, resta, media, calculo de máximo y mínimo y por los operadores lógicos AND (o conjunción) y OR (o disyunción). Tras un estudio de los requisitos de personalización de entornos como el hogar o la oficina, se llegó a la conclusión de que con dichos operadores y operaciones matemáticas todas las posibilidades estaban cubiertas. Las operaciones relacionales permiten comparar dos operandos (condiciones) que pueden ser números, valores alfanuméricos y booleanos, cuyo resultado puede ser verdadero o falso.

  Veamos todo lo expuesto anteriormente con un ejemplo en el que únicamente se evalúa una condición simple, y otro en el que se evalúa una condición compuesta formada por dos condiciones. Imaginemos que el usuario Garbiñe está en casa y va del salón a la cocina. El evento que recibirá el MR una vez Garbiñe haya entrado en la cocina es de la forma:

```
"(a2484ae7-51ce-498b-aa53-00f28baf936e::Servicio-Locator::User = Garbiñe)"
```

El primer valor alfanumérico que aparece es el identificador único del dispositivo de localización de la cocina. A continuación, se identifica el servicio que corresponde a tal evento, 'User' se refiere a la variable que ha cambiado de valor y ha provocado el evento y el último parámetro es el valor de dicha variable.

En el caso de que una regla tenga como condición una condición compuesta veamos qué ocurre. Por ejemplo, pensemos que en la condición de la regla intervienen una variable de un dispositivo y varias variables de tiempo: 'si el usuario Garbiñe se encuentra en el salón en agosto entre las tres y las seis de la tarde, entonces…'.

Una vez definida la regla, el MR guarda la información siguiente:

```
"(AND            (DeviceA_Identifier::Servicio-Locator::User = Garbiñe)
 (AND
(Date >= Month=August;Day=All;Hour=15;Minute=0;)
(Date <= Month=August;Day=All;Hour=18;Minute=0;)
 )
)"
```

Esto corresponde a la información generada por el MR ante tal condición compuesta.

- ✓ '*Rules'* o 'Reglas' constan de la lista de reglas cargadas en el sistema y de la información relativa a las mismas: estado de las condiciones, variables y acciones asociadas. Las reglas pueden estar activadas o no, ya que en el propio motor de reglas existe la posibilidad de cambiar el estado a una regla en función de las necesidades del momento, por ejemplo, pudiendo un administrador activar o desactivar las reglas cargadas en función de las necesidades del momento y del día.
- ✓ El 'Rule Evaluator' es el componente 'inteligente' del sistema. Se encarga de evaluar las reglas afectadas por los cambios de variables, de verificar si existen conflictos entre las reglas afectadas y las de todo el conjunto antes de enviar el evento que disparará la acción de modificación real (física) de variables y de verificar que las actuaciones sobre los dispositivos involucrados se llevan a cabo correctamente.
- ✓ El componente '*Actions*' está formado por la lista de acciones definidas en las reglas. Son utilizados por el 'Action Dispatcher' cuando lo solicita el gestor de reglas. En cuanto a las peculiaridades de las acciones, cabe destacar que el usuario puede definir que si no se ha podido llevar a cabo la actuación, el ejecutor de acciones intente una y otra vez hasta que lo consiga. Existe también una acción bastante peculiar, la acción 'Do Nothing', es decir, una acción para no hacer nada.

Este tipo de acción va a ser muy útil en caso de conflictos y cuando se utiliza en conjunto con una condición de tipo sistema. Por ejemplo, si queremos definir una regla de tipo entorno para que durante la noche no se pueda encender la TV, creamos la condición compuesta con las fechas y la variable a monitorizar 'Servicio-TV::Set_State::ON', con la acción de tipo 'Do Nothing'. En el momento que alguien intente poner la TV en marcha, entrará en conflicto con la regla definida, y al ser de mayor prioridad, la acción a realizar será nula, ya que prevalece sobre cualquier otra. Estas cuestiones se analizarán con más detalle en el siguiente capítulo.

- ✓ Respecto al '*Action Dispatcher'*, se trata del componente que se encarga de ordenar la ejecución de las acciones de las reglas y gestiona si la actuación se ha llevado a cabo correctamente o no en los parámetros definidos por el usuario. Es decir, si el usuario ha decidido que quiere que sea notificado antes de la ejecución de una acción es el ejecutor de acciones el que se encarga de gestionar esta característica, al igual que de reintentar una actuación en caso de no encontrar el dispositivo.

Una vez identificados los componentes que intervienen, vamos a profundizar en los procesos de la carga de reglas y la evaluación en el motor de reglas.

- ➢ Carga de Reglas:

  La incorporación de reglas al motor de reglas se realiza parseando y cargando toda la información contenida en el fichero de reglas generado por el editor de reglas. Son cargadas: la lista de las variables utilizadas en cada condición de las reglas, la lista de las reglas con sus condiciones y las acciones de las reglas. En el momento que los dispositivos comienzan a anunciarse, se realizan las asociaciones necesarias entre las variables físicas y lógicas involucradas en las reglas. Como se muestra en la figura 5.3, todo dispositivo publica sus variables o 'Device Variables', y las pone a disposición del resto en la nube. En el momento que el motor de reglas se pone en marcha, descubre los dispositivos disponibles en la nube, recibe la lista de ellos y crea una estructura lógica paralela suscribiéndose y emparejándose con cada una de las variables. Por otro lado, durante el proceso de la carga de reglas, se han de asociar por cada regla, las acciones que han de ocurrir en caso de satisfacerse la regla, y por otro, como la regla se divide en condiciones, por cada condición, la regla genera una asociación para recibir y ser notificado de que la condición a sufrido variaciones. A su vez, por cada condición, se crean asociaciones con las variables lógicas de los dispositivos que el motor de reglas ha ido descubriendo. Cuando ha ocurrido la asociación de variables, es entonces cuando cualquier cambio en la variable física de un dispositivo, es notificado y comienza el proceso de evaluación.

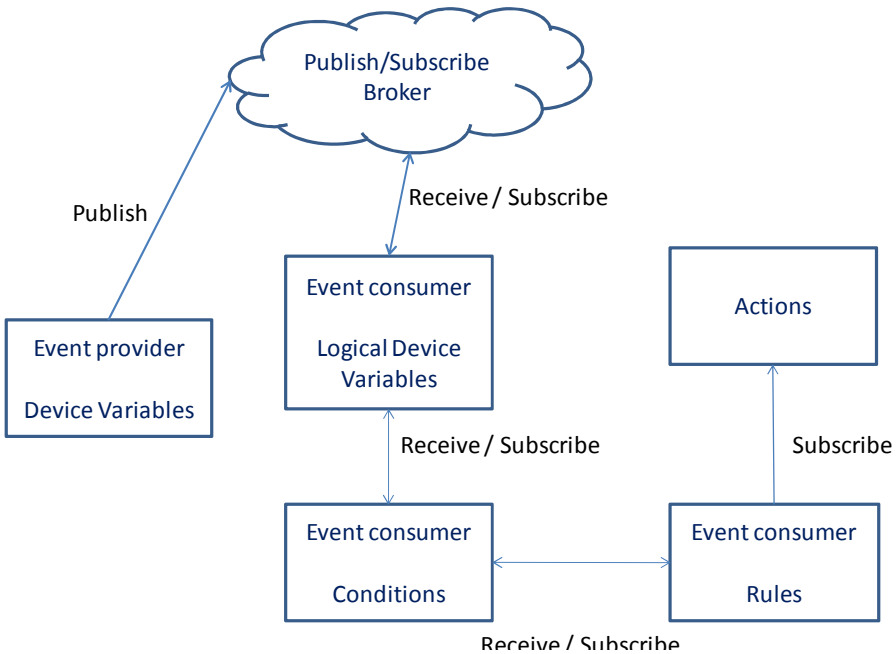


Fig. 5.3 Esquema publicación / suscripción

- Evaluación de Reglas:

Con todo ello, si una 'device variable' sufre cambios, ese cambio es recibido por su homónimo 'logical device variable' del sistema, y si esta 'logical device variable' se encuentra involucrada en una o varias 'conditions', la notificación de modificación hace que se evalúe dicha condición. Si la evaluación de la condición supone un cambio, un evento de cambio de condición es enviado a la regla o 'rules' correspondiente. La regla es evaluada y si es satisfactoria, las 'actions' correspondientes serán ejecutadas. Comentar que si una regla consta de varias condiciones y solo una variable involucrada en una condición es el iniciador de las evaluaciones, únicamente la condición asociada con dicha variable es evaluada. Al no sufrir variaciones las variables del resto de condiciones, como se sabe el estado en el que están, no hay que volver a realizar evaluaciones a nivel de condiciones. En este caso solo se evalúa a nivel de regla.

El MR evalúa condiciones de las reglas y la correspondiente activación de las reglas cuyas condiciones sean verdaderas. La evaluación de las condiciones se lleva a cabo si y sólo si llega un evento de una variable incluida en una de las condiciones. En caso de que se cumpla la condición, se producirá otro evento para que se active la regla correspondiente, y como resultado se ejecutarán las acciones correspondientes. El mecanismo de activación es el mismo que existe para la asociación de las variables lógicas y las condiciones. Las acciones se activarán vía eventos internos, sin que haya que revisar toda la lista de acciones hasta encontrar la necesaria. Por una parte, no se utilizan las estructuras clásicas tipo *switch-case o IF* encadenados, por otra no hay una selección previa de las acciones, ya que la ejecución vendrá dada al haber emparejamiento entre la regla y sus acciones. Por un lado, existe una lista global de acciones y, por otro, cada

regla mantiene una lista con las referencias a sus acciones que se encuentran en la lista global. De esta manera se evitan duplicidades y el acceso a las acciones de una regla es inmediato, no hay que buscar por toda la lista global de acciones sino que se accede directamente desde la lista privada de acciones de cada regla.

Lo comentado anteriormente implica que, “por cada notificación de cambio de variable se va a crear un espacio de ejecución independiente, es decir un hilo independiente que se encargará de procesar y seguir con la secuencia de ejecución, pasando los eventos estipulados entre los diferentes objetos (variable, condición, regla, acción) para su procesamiento y evaluación”. Dicho de otro modo, en el MR simultáneamente pueden estar evaluándose varias condiciones y/o ejecutándose varias acciones de las reglas, que son el resultado de haber recibido diferentes notificaciones de cambios de una misma variable o varias variables diferentes, prácticamente en el mismo momento. De hecho, para que el motor de reglas pueda soportar la llegada de varios eventos de una misma variable en casi tiempo real (la peor situación con la que se puede encontrar el motor de reglas), se han utilizado colas para que la llegada de una variable no disturbe la ejecución de la variable inmediatamente anterior. Utilizando el mecanismo de colas, garantizamos que a la llegada de los eventos queden almacenados, no se pierdan y que sean procesados cuando la CPU pueda, es decir, tan pronto cuando termine de lanzar la ejecución de la variable anterior. Esto implica que el tiempo de respuesta del motor de reglas dependerá del hardware utilizado para su ejecución. Más adelante en el capítulo de experimentación y validación se analiza este aspecto referido al rendimiento.

La ejecución de las acciones se realiza en el orden preestablecido por el usuario en el momento de la creación de las mismas. Es decir, por ejemplo para poder encender solamente la lámpara de la mesilla de noche, primero tendremos que asegurarnos que todas las lámparas se encuentran apagadas en la habitación, incluyendo la de la mesilla de noche, y después encender la de la mesilla. Esto resulta de vital importancia cuando realizamos tareas y acciones sobre dispositivos por grupos.

Si una regla se ha satisfecho, antes de proceder con la ejecución de su acción o acciones correspondientes se verifica que la variable no tenga ya el valor deseado. La comprobación se realiza sobre el valor de la variable lógica del sistema, ya que mantiene el mismo estado del valor real de la variable del dispositivo. De esta manera optimizamos el tiempo de ejecución y evitamos generar tráfico innecesario (y la orden de modificación de la propia variable) por la ejecución de la acción, con el consiguiente ahorro de envío innecesario de mensaje al dispositivo. No se trata más que de evitar el envío del comando al dispositivo lámpara para la acción de encender la luz, si la luz ya está anteriormente encendida.

Por otro lado, se puede elegir, dependiendo de la confiabilidad del sistema, entre una ejecución síncrona o asíncrona en la función de llamada para establecimiento del valor de la variable en el dispositivo. Esto es, en una comunicación síncrona, se espera hasta tener la confirmación de que la orden o instrucción se ha ejecutado. En cambio, en una comunicación asíncrona, no se tiene confirmación de si la ejecución se ha llevado a cabo o no, simplemente se envía el comando a través de la red y se supone que el mensaje llega a su destinatario y éste es capaz de ejecutarlo. También se puede definir un tiempo de espera después de la ejecución de una acción, para el caso de que existiera algún dispositivo con un tiempo de respuesta lento, y así esperar a la completa ejecución de una acción y a los cambios de variable o variables que conlleven antes de seguir con la ejecución del resto de acciones.

5.4.2 Características del motor de reglas

El motor de reglas juega un papel clave para lograr la personalización de un entorno. Básicamente, el motor de reglas tiene dos funciones (1) vincular o incorporar las reglas definidas a través del editor y (2) evaluar y ejecutar dichas reglas. A continuación se van a destacar algunos aspectos relativos al funcionamiento del motor de reglas.

5.4.2.1 *Resolución de conflictos*

El MR implementa un mecanismo para la resolución de conflictos. Como se estudió en el estado del arte, la mayoría de los trabajos utilizan prioridades para determinar que regla tiene que ser activada en caso de conflicto. En nuestro caso los conflictos se resuelven comprobando la prioridad asociada a cada una de las reglas que entran en conflicto. Tomando como referencia las prioridades, el motor de reglas evalúa si la acción se puede llevar a cabo o entra en conflicto con una o varias reglas, alguna de ellas con prioridad más alta. El MR propuesto siempre evalúa la regla cuya condición ha sido satisfecha al producirse un evento y, a continuación, verifica si las acciones a ejecutar entran en conflicto con alguna de las reglas activadas con la misma acción o acciones definidas. Una vez detectadas las reglas en que las acciones son coincidentes, se evalúan las reglas en conflicto y el valor aplicado es aquel perteneciente a la regla con mayor prioridad en caso de conflicto.

Supongamos que un padre y un hijo han definido dos reglas para que se encienda la televisión en un canal predeterminado. La prioridad de la regla definida por el padre (R1) es mayor que la prioridad de la regla definida por el hijo (R2). R1 especifica que cuando el padre entre en el salón la televisión ponga el canal 1. Y la regla R2 expone que la televisión ponga el canal 5. Imaginemos que el padre está en el salón viendo el canal 1, tal y como se muestra en la figura 5.4, el hijo entra en la sala y el MR recibe el evento de que el hijo ha entrado en la sala. Entonces, se evalúa la condición de la regla R2 y se cumple, por lo que se debería cambiar el canal. Sin embargo, antes de realizar el cambio de canal, el mecanismo de búsqueda de resolución de conflictos se activa, y el evaluador de reglas busca en la lista global de acciones las coincidentes a la que se pretende modificar e identifica las reglas a las que pertenecen, por lo que en nuestro caso detecta que la regla R1 tiene la misma acción que la acción de la regla

R2. En consecuencia, el MR lleva a cabo la evaluación de la regla del padre R1, que se cumple en este caso, por lo que la R2 queda invalidada, es decir, el valor utilizado es el que marca R1. En este caso ocurre que el valor de la variable (canal de televisión) ya está establecido, y como el ejecutor de acciones antes de cambiar una variable verifica que el valor deba ser diferente, se termina el hilo de ejecución producido por la entrada del hijo en la sala o la regla R2.

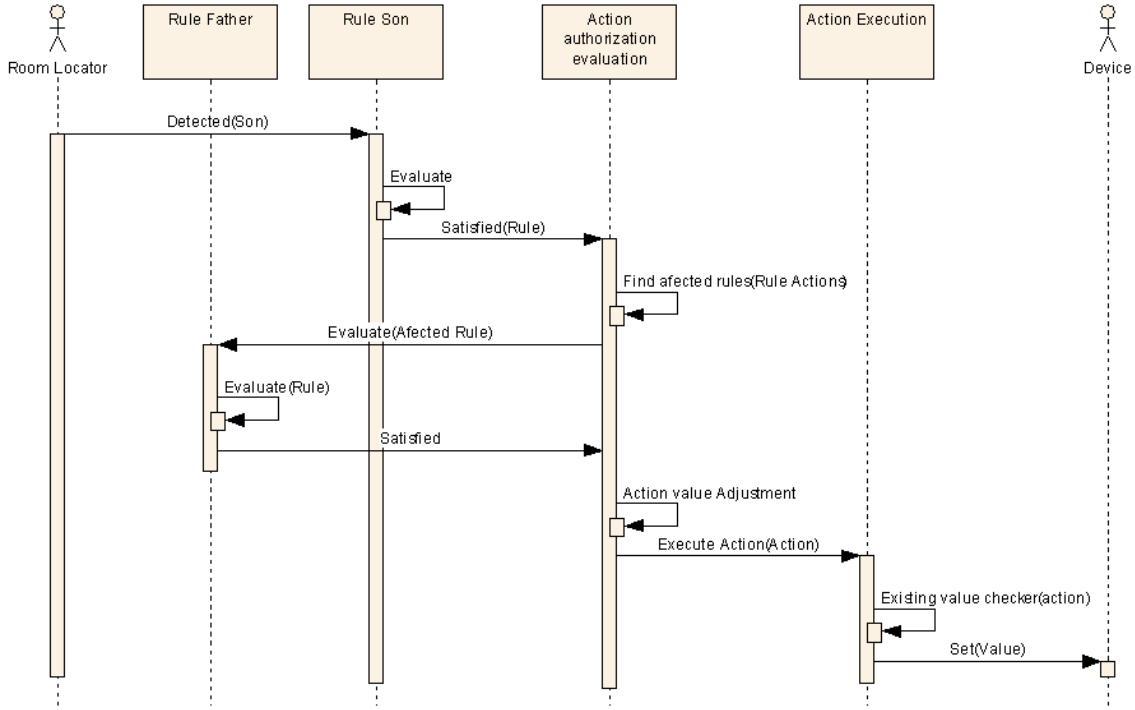


Fig. 5.4 Proceso evaluación de prioridades de la acción

Existe un tipo de reglas denominado de ‘sistema’ que únicamente se evalúa si entran en conflicto con las acciones definidas por un usuario en el momento de la evaluación. Las reglas de tipo sistema se usan para establecer límites por el administrador, es decir, si por ejemplo no queremos que la temperatura de consigna sea mayor que 23ºC, definiremos una regla de este tipo, y siempre que alguien declare una acción con temperatura mayor que 23ºC, entrará en conflicto con esta regla de entorno definida y siguiendo el mismo mecanismo de resolución de prioridad, el valor que prevalece es el que marca la variable de la regla del sistema.

El proceso de evaluación de prioridades primero buscará coincidencias entre las acciones de las reglas activadas, segundo con el resto de acciones de las reglas cargadas en el motor de reglas que no sean reglas del sistema, y finalmente, buscará coincidencias con las reglas del sistema.

5.4.3 Despliegue

La instalación del MR se podría llevar a cabo de varias formas. Dependiendo de las características del escenario y de los dispositivos que intervengan en el sistema,

podríamos pensar en al menos dos tipos de distribuciones, una de ellas sería siguiendo una arquitectura orquestada y la otra una arquitectura coreografiada.

5.4.3.1 Orquestación

La arquitectura tipo orquestación es aquella en la que existe un único MR, y es este MR el que tiene todo el conocimiento y el grupo de reglas para que en cada momento pueda detectar y responder a los diferentes cambios de las variables de los dispositivos. Cualquier otro dispositivo en esta configuración no es más que un elemento pasivo que es capaz de informar y realizar los cambios que desde fuera se le indican. Toda la 'inteligencia' se encuentra concentrada en el único dispositivo, el que ejecuta el MR.

5.4.3.2 Coreografía

La arquitectura coreografiada es aquella en la que varios dispositivos albergan su propio MR. De esta manera, si ejecutamos una serie de reglas específicas para cada caso, haremos que el dispositivo reaccione por sí mismo ante cambios producidos en el sistema. En este tipo de arquitectura, la 'inteligencia' se encuentra distribuida en cada elemento o dispositivo de la red, funcionando cada uno de ellos de manera independiente del resto. Utilizando este tipo de distribución, podemos cambiar el comportamiento de los propios dispositivos, no solo haciendo que modifiquen ciertos valores, sino que, en función de las necesidades del momento, se comporten de una manera o de otra, pasando de ser un dispositivo pasivo a un dispositivo reactivo que se adapta a las necesidades y requisitos solicitados. Entonces es posible que varios MR gestionen las reglas en un mismo entorno, ahora bien, aunque queda fuera de los objetivos de este trabajo de tesis, podría ocurrir que se llegaran a situaciones contradictorias, dado que la resolución entre conflictos entre varios MR no se ha llevado a cabo.

Las dos arquitecturas de despliegue pueden llegar a ser válidas. La administración y gestión de un sistema tipo orquestación parece a-priori más sencilla. Por otra parte, en una red poco robusta parece que una arquitectura coreografiada puede adaptarse mejor a los requisitos del entorno, y reaccionar mejor frente a los fallos de red, desconexiones, etcétera, ya que se pueden distribuir los MR de forma que cubran los fallos de sistema que pudieran ocurrir. De todas formas, los requisitos de computación no serán comparables al dispositivo que alberga el MR con todas las reglas de un escenario o ubicación. Debido a la necesidad de tener que dar respuesta a cientos de reglas en el caso orquestado y a una decena (como mucho) en el caso coreografiado, habrá que tener en cuenta esta circunstancia a la hora de la selección de las características del dispositivo en el que se vaya a ejecutar el motor de reglas. Por ello, dependiendo de cada instalación y características del escenario de aplicación, habrá que optar por una de las arquitecturas o una mezcla entre ambas.

De todas formas, no está extendida la utilización de dispositivos que incluyan un motor de reglas. En [Jung+07] por ejemplo justifican la utilización de motores de reglas en

cada uno de los dispositivos. En este caso el número de reglas de las que cada dispositivo sería responsable no debería ser muy amplio. Otra alternativa es la de un PC o PDA con ciertos recursos para realizar labores de control. En nuestro caso, al dispositivo simulador que se ejecuta sobre PC se le ha añadido la funcionalidad del motor de reglas, así todo dispositivo puede ser MR y formar parte de la red coreografiada que anteriormente se ha mencionado. En remoto se puede habilitar o deshabilitar esta funcionalidad del motor de reglas, y a su vez, modificar y cargar ficheros de reglas que se ejecutarían en dicho motor de reglas (véase anexo A), pudiendo fácilmente crear una red de motores de reglas en el mismo ámbito de ejecución.

5.4.4 Otros mecanismos

En entornos tan cambiantes y a veces poco robustos donde las comunicaciones, normalmente inalámbricas, hoy en día pueden fallar, el conocimiento del estado de los dispositivos conectados a una red es imprescindible. Para detectar este tipo de estados anómalos debido a desconexiones o apagado fortuito, se han añadido varios mecanismos para tratar de solventarlos. Uno de ellos, el '*Ping*', va dirigido a los dispositivos, y el otro, el '*isAlive*', a garantizar un estado siempre consistente de las variables involucradas en las condiciones.

5.4.4.1 Ping

Mediante este mecanismo de 'ping', periódicamente se verifica que un dispositivo sigue vivo y al alcance. En caso de que no hubiera respuesta al ping, el sistema borra de su lista de dispositivos activos el dispositivo que se queda sin conexión y todos los elementos asociados con dicho dispositivo dentro del motor de reglas se de-suscriben partiendo de cada una de las variables del dispositivo ausente y eliminando toda la información que hace referencia al mismo. Si el dispositivo volviera a conectarse a la red, el MR volvería a realizar todo el proceso de suscripción, tal y como si fuera la primera vez.

La forma de proceder es la siguiente: según el tiempo definido de un minuto, se realiza un barrido a todos los dispositivos que en teoría están conectados. El barrido consiste en una llamada síncrona a la función de *ping* que generalmente incorporan todos los dispositivos. Si la llamada síncrona falla es porque no está al alcance, entonces para asegurarse de que el motor de reglas, por ejemplo no invoque una acción de un dispositivo que no está conectado, se elimina toda referencia al dispositivo y sus variables.

5.4.4.2 IsAlive

Periódicamente, cada variable lógica verificará si su correspondiente variable real está viva (activa). Si no lo estuviera (denominada pasiva), procederá a auto-asignarle el valor por defecto preestablecido (si existe) o asignará el último valor recibido de dicha variable, dependiendo de la opción que el usuario haya tomado al definir la regla.

Mediante el mecanismo de 'IsAlive', garantizamos que toda variable involucrada en una condición siempre tenga un valor. Es decir, a pesar de que las variables involucradas en una condición de un dispositivo no estén al alcance, siempre van a tener un valor al menos por defecto. A la hora de crear las condiciones, el usuario podrá elegir entre utilizar un valor por defecto para los casos en los que el dispositivo no se encuentre disponible o que se utilice el último valor conocido. De esta manera, garantizamos que una regla compuesta de varias condiciones, que dependen de varios dispositivos, no falle en caso de que alguno de los dispositivos no esté disponible.

Habrá situaciones en que lo conveniente sea, por seguridad, pasar y utilizar cierto valor por defecto, y en otras seguir con el último valor conocido.

Los mecanismos de 'ping' e 'IsAlive' son complementarios. Con uno de los mecanismos conocemos si el dispositivo está ausente o no, y con el otro si las variables de un dispositivo están respondiendo bien o mal.

5.4.4.3 Notificaciones

Dependiendo de la opción seleccionada por el usuario al crear una regla, antes y después de la ejecución de una regla, el usuario puede ser informado de que una regla (la acción de la regla) está a punto de ejecutarse, y solicitar información de confirmación para continuar o parar la ejecución de las acciones y, por otro lado, puede ser informado del estado de ejecución de las acciones.

5.4.4.4 Personalización del reloj para la simulación

El MR dispone de una utilidad para poder personalizar el reloj que usa de referencia. Aunque normalmente este reloj será igual al del PC o dispositivo en el que se ejecuta, puede que, por ejemplo, para realizar simulaciones, sea interesante utilizar otra referencia de tiempo. La realización de simulaciones sincronizadas con un período de tiempo concreto, menor que el tiempo actual, puede ser interesante. En este sentido, por ejemplo, el poder simular las acciones que ocurrirían en 24 horas, en solo 24 minutos es interesante desde el punto de vista del tiempo que se ahorra en cada simulación. En el fondo, lo que ocurre es que aceleramos el reloj utilizado como referencia en función del tipo de simulación que queremos llevar a cabo.

5.5 Conclusiones

Hoy en día, la idea de que los usuarios configuren sus entornos parece viable, necesaria y útil. Viable por la existente y creciente variedad de dispositivos (esto es, dispositivos empotrados, embarcados o embebidos) que hay en los hogares y que van a permitir crear comportamientos ambientales configurables. Una idea necesaria y útil porque cada vez más se ve la necesidad de que las personas puedan acceder y controlar cierto tipo de información dónde quiera que estén, y disfrutar de nuevos servicios relacionados con su localización, hora y entorno que les rodea.

En este sentido, creemos que la gente necesita permanentemente cambiar y rediseñar el comportamiento del entorno que le rodea para ajustarse a las necesidades y preferencias del momento. Es por ello que el sistema basado en reglas que se ha descrito en este capítulo puede ayudar en esta tarea. El sistema completo constará de un motor de reglas (que detecta eventos y ejecuta reglas) y de un editor de reglas para la generación de las mismas por cualquier tipo de usuario. En el siguiente capítulo 6 se aborda la temática de la personalización del entorno más cercano y circundante por parte del usuario final y la creación de reglas utilizando un editor de reglas.

Capítulo 6: Personalización de entornos

Las características de nuestros hogares, sitios de trabajo e infraestructuras están cambiando con las nuevas tecnologías de la información y del conocimiento y se están empezando a introducir nuevos elementos que proveen servicios para, por una parte, mejorar las capacidades del usuario y por otra su calidad de vida. Sin embargo, la complejidad de crear comportamientos ambientales personalizados, o en general servicios personalizados, está creciendo muy rápido con la llegada y el uso de una gran variedad de dispositivos (como son los dispositivos embarcados) junto con una funcionalidad cada vez mayor. Este gran número de funcionalidades embarcadas y las combinaciones de éstas ofrecen al usuario un sin fin de nuevos servicios en su entorno. No obstante, por mucha tecnología que instalemos en nuestros hábitats y entornos de trabajo, sino se permite al usuario "educar" o "personalizar" el comportamiento en conjunto de estos dispositivos, se cree que nunca se podrán satisfacer las necesidades finales del usuario.

En este capítulo se describe el editor de reglas que permitirá la personalización de un entorno, como el hogar o una oficina de una manera sencilla e intuitiva.

6.1 Introducción

El Editor de Reglas (ER) será la herramienta a utilizar por cualquier persona para lograr la personalización de un entorno debidamente sensorizado. A través del ER, se mostrarán todos los dispositivos y servicios disponibles al alcance del usuario. De acuerdo con estos dispositivos descubiertos y sus servicios disponibles, el usuario creará el conjunto de reglas de personalización que posteriormente serán ejecutadas por el Motor de Reglas (MR). Al igual que el MR, el ER se basa en los mecanismos de interoperabilidad y descubrimiento descritos en el Capítulo 3. El objetivo principal del ER es facilitar al usuario la creación de las reglas, aunque está provisto de otra serie de utilidades para realizar tareas como, por ejemplo, las simulaciones de las acciones.

TAMAmI (*TAilor-Made Ambient Intelligence)* es el nombre dado al ER. A continuación, se detallan las características que debería tener TAMAmI y una descripción breve del mismo.

6.1.1 Requisitos para la Personalización de un Entorno

Tras un análisis exhaustivo del estado del arte en este dominio, como se vio en el apartado 2.3, se identificaron las características fundamentales que debe cumplir un *middleware* para lograr que los entornos sean configurables y para que los usuarios puedan configurarlos. Estas características vienen definidas por los siguientes retos:

- Descubrimiento y enlazamiento (*binding*) de cada servicio ubicuo: mecanismos para descubrir servicios y definir la invocación de los mismos.
- Coordinación y composición de servicios: situación que se presenta cuando la acción definida por una regla se alcanza al componer varios servicios.
- Naturaleza dinámica de las reglas: las reglas vienen definidas por los eventos, condiciones y acciones, y estos parámetros cambian cuando aparecen / desaparecen ciertos dispositivos. Además, un usuario también podría cambiar en tiempo de ejecución las reglas definidas.
- Principio de consistencia con las reglas existentes: las nuevas reglas creadas por el usuario sean consecuentes con las reglas ya existentes.
- Resolución de conflictos: un conflicto se puede dar cuando las reglas definidas por usuarios distintos controlan el mismo dispositivo de una manera diferente.
- Restricciones de tiempo: el problema viene al definir los eventos temporales (puntuales o en un periodo de tiempo) y el comportamiento dependiente del tiempo al invocar una regla.
- Decisión del usuario: permitir al usuario la toma de decisiones en los casos oportunos.
- Preguntar al usuario: ofrecer la posibilidad de notificar y preguntar al usuario sobre las diferentes reglas antes de que se desencadene la acción definida.

Algunos de estos retos ya han sido abordados en el capítulo anterior por el MR; sin embargo, otros deberán ser abordados en el ER. Aparte de estas características descritas hay que tener en cuenta otros aspectos, que siendo también importantes, están más relacionados con la toma de una decisión ante un problema específico. Por ejemplo, estos pueden ser el manejo de las acciones concurrentes o tener en cuenta el orden de ejecución de los eventos.

6.1.2 TAMAmI: Un Entorno para la personalización

El editor TAMAmI básicamente ha de cumplir con dos objetivos:

I. Que sirva de herramienta de configuración, administración y gestión del entorno, de manera que una persona con un perfil de administrador pueda conocer la situación actual de los dispositivos, al igual que pueda identificar cualquier problema de comunicación actual o pasado, para poder solucionar posibles problemas.

II. Que sea un entorno de configuración amigable para el resto de usuarios, donde cómodamente puedan definir o transformar sus preferencias en forma de reglas.

A continuación se mencionan las principales funcionalidades identificadas para dar cobijo a los requisitos detectados.

6.1.2.1 Gestión de usuarios

Todo sistema de administración donde estén involucrados varios usuarios deberá contar con algún mecanismo de gestión de usuarios. La gestión de usuarios permitirá definir los permisos para acceder a las funciones del sistema y asignar las prioridades a cada usuario para los casos en los que haya.

6.1.2.2 Lista de dispositivos

Para el fácil manejo de los dispositivos disponibles en el entorno, es aconsejable que el configurador de reglas muestre una lista de los dispositivos descubiertos en la red, con los servicios asociados disponibles. De esta manera, de un vistazo el usuario será consciente de la red de dispositivos a su alcance con sus posibilidades de personalización.

6.1.2.3 Creación de las reglas

La parte de edición o creación de reglas será la parte nuclear o más importante del sistema, en el que utilizando los dispositivos descubiertos y de acuerdo con las necesidades o deseos de personalización del usuario, se definirán las condiciones y las acciones de las reglas. Más adelante en este capítulo en el apartado 6.1.3 se puede encontrar información extendida relativa a la creación de las reglas.

Tipos de reglas

Se van a poder definir tres tipos de reglas: '*ordinarias*', '*sistema*' y '*grupo*'. Inicialmente con los tipos de reglas definidos se estima que se pueden crear todo tipo de condiciones y cubrir las necesidades planteadas y definidas a lo largo de este trabajo de tesis.

I. Las reglas 'ordinarias' son aquellas reglas que los usuarios definen y que se evalúan cada vez que alguna de las variables involucradas ha sufrido modificaciones.

II. Las reglas de tipo *sistema* no se evalúan cuando las variables cambian de valor, sino que se utilizan para establecer límites, restricciones a las acciones definidas por los usuarios o solucionar conflictos. Se utilizan para fijar límites o restricciones a las acciones definidas por los usuarios. Se evalúan después de que una regla ordinaria se haya satisfecho para comprobar si la acción a realizar tiene restricciones y puede ser o no validada. Al igual que las reglas ordinarias este tipo de reglas se cargan en cada MR y prevalecen sobre todo el resto de reglas. Lo que se persigue con esto es que el administrador defina una serie de reglas de sistema que acoten el comportamiento de una estancia u hogar, independientemente del resto de reglas de los usuarios.

    Por ejemplo, se puede definir una regla de sistema tal que "entre las dos y las cuatro de la tarde el volumen de la televisión de la sala no sea mayor que el 30%", ya que puede haber gente que se encuentre en la siesta. Por otro lado, la abuela puede tener una regla para poner el volumen de la televisión al 70%

cada vez que entra en la sala. Siempre que el MR vaya a ejecutar la regla de la abuela para ajustar el volumen de la televisión entre las dos y las cuatro de la tarde el valor que utilizará será el de 30% en vez de 70% que correspondería a la regla de la abuela, ya que existe la regla del sistema que la limita. El MR detectará el conflicto entre la regla ordinaria del usuario y la del sistema en el momento de la ejecución de las acciones y aplicará el valor de la regla del sistema si se satisface la condición involucrada.

Es decir, las reglas de sistema se evalúan o se tienen en cuenta antes de realizar una modificación en las acciones, de alguna manera sirven para validar las acciones a realizar por el sistema.

III. Y, finalmente, las de tipo *grupo* son aquellas reglas que se encargan de agrupar dispositivos por funcionalidad, de manera que podemos agrupar, por ejemplo, todos los dispositivos que ofrecen el servicio de luz por habitaciones. Las reglas de tipo grupo son reglas cuyas condiciones y acciones están asociadas a un grupo de dispositivos con ciertas características comunes (generalmente que ofrecen un mismo servicio). Una regla de tipo grupo no define las condiciones ni las acciones de la manera convencional, sino que se definen de tal manera que el MR identificará al dispositivo o a los dispositivos involucrados y las acciones a invocar de los mismos (por ejemplo agrupar todos los dispositivos que ofrecen 'luz', para poder encender/apagar con una sola instrucción). El MR primero analizará los dispositivos cuyas variables cumplen la condición de la regla de tipo grupo. A continuación, el MR creará la lista de acciones asociadas a cada dispositivo identificado. Para ello buscará la correspondencia de acciones con las definidas previamente en la regla de tipo grupo. Estas reglas sirven también para actuar sobre dispositivos que inicialmente no habrían sido identificados en el sistema.

   Se debe notar que puede ocurrir que el cumplimiento de la parte condicional de una regla de tipo grupo no implique la ejecución de ninguna acción. La razón es que primero han de localizarse los dispositivos para poder invocar sus acciones. Los dispositivos implicados en una regla de tipo grupo se irán agrupando para poder ser utilizados posteriormente en otras reglas como dispositivos destinatarios de la acción pertinente. La dinamicidad de una regla de tipo grupo se puede verificar con detalle en la ejecución en el MR (Anexo B en el apartado 4). Es en el MR donde se observa cuando la regla es satisfecha y, como en consecuencia, un dispositivo es añadido a la lista dinámicamente.

Tipos de Condiciones

Las condiciones que el usuario va a poder crear usando el editor TAMAmI pueden ser de cuatro tipos: '*simples u ordinarias'*, '*temporales'*, '*calculadas'* y '*de grupo*'. Para definir las condiciones, se usan los operadores matemáticos '=, <,>, <=, >=', que permiten operar con el valor de las variables de los dispositivos a controlar. Utilizando

los operadores lógicos AND y OR, y operandos matemáticos como Suma, Resta, Media, Máximo y Mínimo, el usuario podrá crear condiciones compuestas.

I. Condición *simple u ordinaria*, es aquella que queda definida por una única variable. Para ello, primero se selecciona el dispositivo que la contiene, el servicio que nos da el valor de la variable y, a continuación, se le asigna el valor que se evaluará.

   Por ejemplo, una regla simple sería “encender las luces de la habitación de la abuela cuando entre por la puerta”. Más adelante en el apartado 6.2 “Edición de Condiciones” se describen más detalladamente este y el resto de los casos de uso expuestos a continuación.

II. Una condición *temporal* es aquella que define un periodo de tiempo en el que la regla podrá ser validada. Este periodo de tiempo vendrá definido por mes, día, hora y minuto.

   Por ejemplo, si queremos que durante la hora de la siesta no se pueda poner el volumen de la televisión más alto que la mitad de su capacidad, crearemos una regla cuyas condiciones serán temporales.

III. La condición *calculada* es una condición que incluye al menos dos variables de distintos dispositivos y cuyo valor es el resultado de aplicar las operaciones de Máximo, Mínimo, Media, Suma o Resta a dichas variables.

IV. La condición sin dispositivo o condición *de grupo* es una condición especial. Gracias al mecanismo definido, mediante esta condición se evaluarán las variables involucradas de dispositivos que pudieran estar conectados a la red en el momento de su creación o bien que se unan a la red a posteriori. Se trata de un mecanismo para generalizar y agrupar los dispositivos por los servicios que ofrecen.

   Por ejemplo, para crear una regla que apague todas las luces de la sala, en la creación clásica de las condiciones se necesita saber qué lámparas o luces existen previamente en la sala. Pero si se utiliza el mecanismo de condición sin dispositivo, se podrán localizar los dispositivos que ofrecen el servicio de luz, y que se encuentran en la sala dinámicamente (en tiempo de evaluación), los actualmente existentes y los que en el futuro se vayan incorporando. El MR será el encargado de interpretar y evaluar en cada caso qué dispositivos no conocidos en el momento de la generación de la regla, cumplen la condición que se ha descrito. También tiene sentido la utilización de este tipo de condición en la generación de reglas ordinarias, ya que dinámicamente se estarán añadiendo los dispositivos que según se descubran se convertirán en parte activa a la hora de la evaluación de la condición.

Se podrán mezclar las condiciones de cualquier tipo para crear condiciones complejas o compuestas dependientes del tiempo y de valores de dispositivos.

Tipos de Acciones

Para modificar los valores de las variables en los dispositivos, se han identificado tres tipos de acciones: '*normal*', '*Do Nothing*' y '*vinculadas'*.

I. Las acciones '*normales'*, son aquellas que el usuario selecciona al crear una regla de la lista de dispositivos disponibles en su entorno, de tal manera que ha de indicar el valor a asignar a la variable concreta del dispositivo.

II. Las acciones de tipo '*Do Nothing'* sirven para denegar permisos o parar la ejecución de las acciones de otra regla. Su función es la de que en caso de conflicto, al tener normalmente mayor prioridad que otra regla, esto permite parar la ejecución de las acciones de la regla con la que entra en conflicto.

III. Las acciones de tipo '*vinculadas'* son aquellas en las que el valor que se le asigna a una variable depende o es subordinada de otra. Es decir, en tiempo de ejecución es cuando se asigna el valor de la variable del dispositivo origen a la variable deseada.

6.1.2.4 Gestión de las reglas

Respecto a las reglas, el sistema en cuestión deberá de proveer los mecanismos necesarios para abrir, modificar, añadir y guardar reglas o listas de reglas. Por lo tanto, el propio editor TAMAmI (figura 6.1) contará con funciones específicas para la gestión de las reglas. Más adelante, se describirá dicho editor con más detalle.

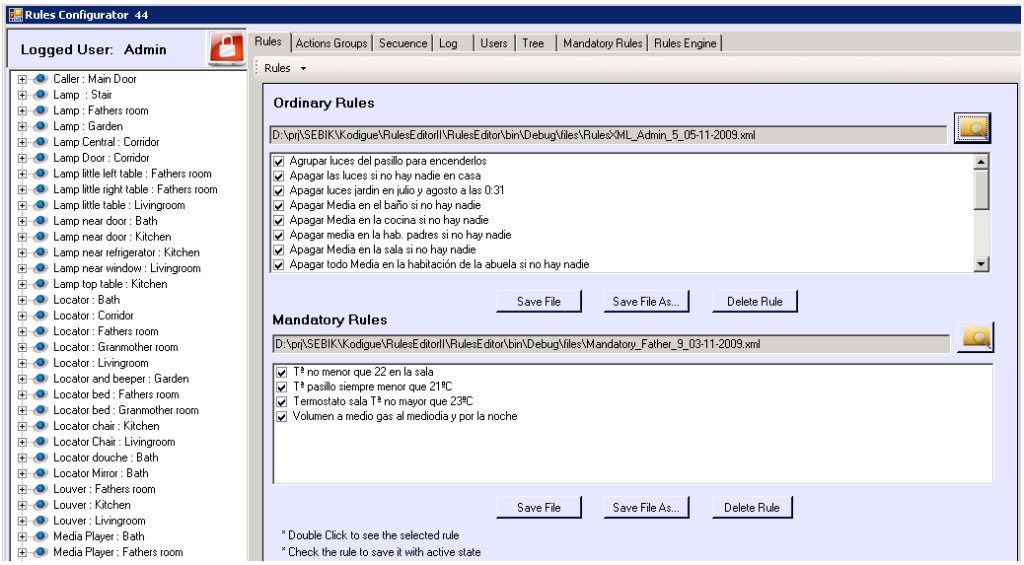


Fig. 6.1 Vista principal del editor TAMAmI

6.1.2.5 Notificador de conflictos

Existe la posibilidad de que un mismo usuario cree reglas que desencadenen acciones contradictorias. Para evitar este tipo de situaciones, el editor de reglas cuenta con un mecanismo de verificación de reglas que avisa al usuario que tal vez esté creando una

regla contraría a otra ya existente. En la figura 6.2 se muestra el tipo de notificación que recibe el usuario en caso de detección de un posible conflicto.

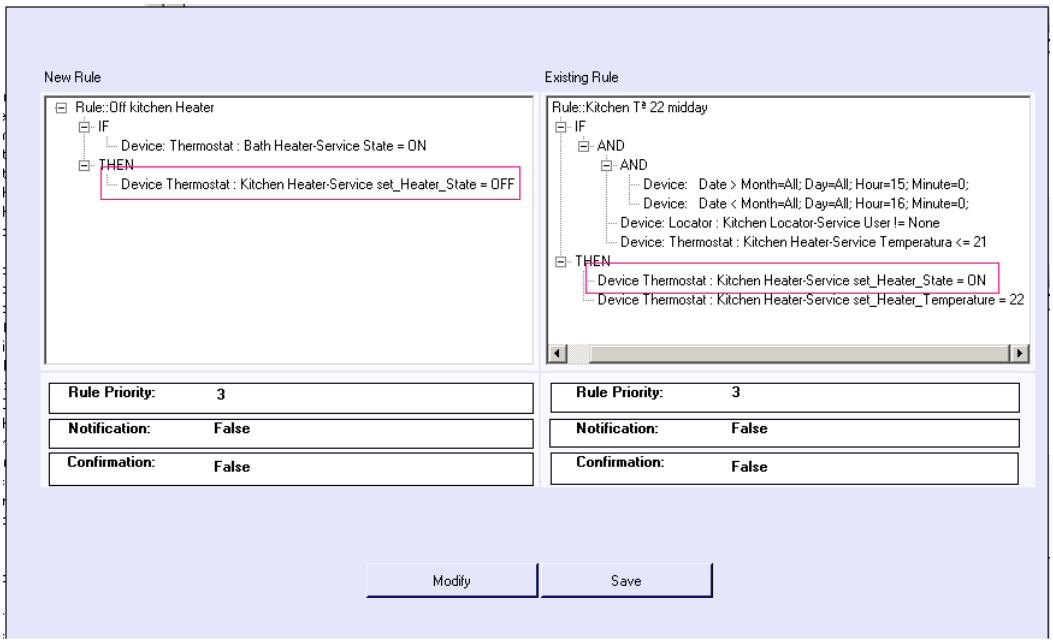


Fig. 6.2 Notificador de posibles conflictos

El sistema únicamente avisa al usuario de posibles actuaciones contradictorias, siendo la responsabilidad del usuario la de decidir si son reglas contradictorias o no.

6.1.2.6 Gestor de motores de reglas

El ER incorpora una utilidad para gestionar los dispositivos que tienen embebida la funcionalidad del MR (véase anexo A Dispositivos virtuales UPnP para más detalle). Se trata de una herramienta para detectar y configurar remotamente los dispositivos que incorporan un MR que en un mismo entorno pueden existir ejecutándose en diversos dispositivos. El aspecto que tiene en el editor de reglas se muestra en la figura 6.3.

Mediante esta utilidad, el ER se convierte en una especie de centro de control para dispositivos que incorporan un MR, de tal forma que desde una ubicación centralizada se puedan controlar y manipular el comportamiento de cada uno de los MR incorporados en los dispositivos. Una vez creadas y guardadas las reglas, se podrán gestionar desde el ER, cargando y activándolas remotamente en los dispositivos. Se trata de una herramienta que facilita la labor de instalación y mantenimiento de los MRs que podrían estar configurados siguiendo una arquitectura coreografiada en el hogar. Esto quiere decir que pueden coexistir varios MR en un mismo entorno y que es el usuario el que decide que reglas se ejecutan en cada uno de los dispositivos dotados con un MR.

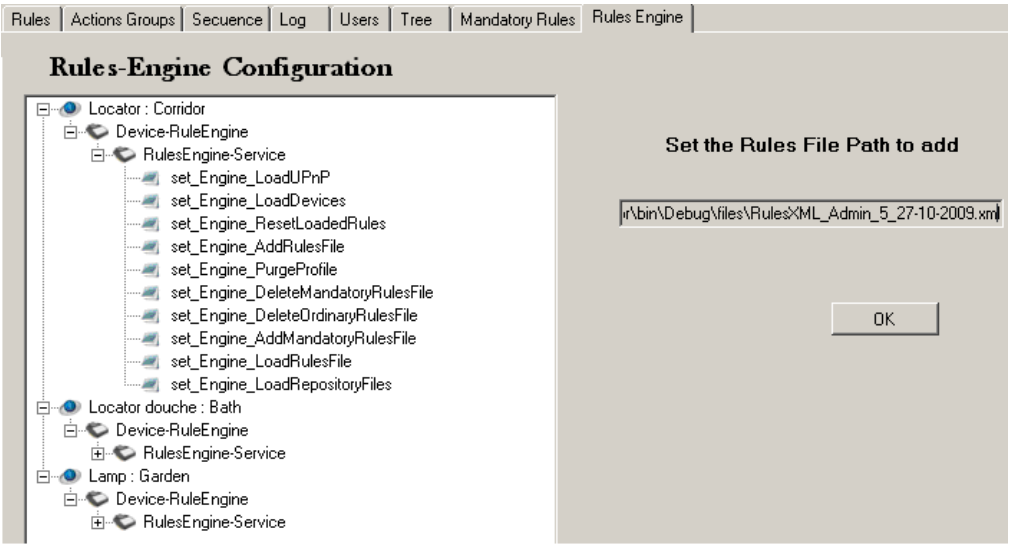


Fig. 6.3 Imagen opciones gestor de motor de reglas

6.1.2.7 Varias opciones

Existe también la posibilidad de recibir notificaciones de ocurrencia de ciertos sucesos. La figura 6.4 muestra las opciones de notificaciones. Resaltar que el MR será un componente que no necesitará de intervención de un usuario para su funcionamiento. No obstante, para ciertos casos, puede ser interesante notificar al usuario de la ocurrencia de cierto evento, por ejemplo si cierto acelerómetro ha detectado la caída de una persona. La notificación podría llevarse a cabo, enviando un mensaje corto a un móvil predefinido, mostrando un mensaje en todas las televisiones (pantallas) en marcha, o diciendo cierto mensaje grabado por los altavoces.

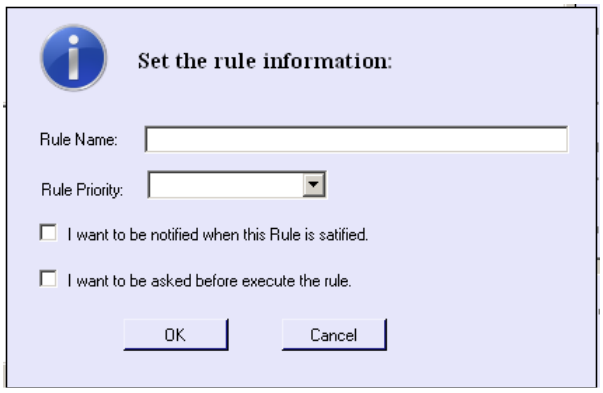


Fig. 6.4 Asignación de nombre, notificaciones y prioridades

La figura 6.4 muestra la ventana de notificación que aparece al crear una regla. A través de esta ventana, se puede establecer:

A. Por un lado, que el usuario sea notificado cuando se vayan a ejecutar las acciones de dicha regla.

B. Por otro lado, que el usuario sea notificado cuando las acciones de la regla se hayan ejecutado correctamente.

6.1.3 Componentes del editor de personalización TAMAmI

Este apartado describe el modus operandi de TAMAmI. Como se va a poder ver, en los siguientes sub-apartados, su implementación ha seguido los requisitos funcionales y no funcionales enumerados en el apartado 6.1.1. Una vez descritas las funciones más generales de TAMAmI, se mostrarán con la ayuda de diferentes casos de usuario que usa este editor, haciendo hincapié en el potencial en cuanto a su usabilidad y capacidad de configuración de dispositivos.

En la siguiente figura 6.5 se muestra una imagen del menú principal del editor TAMAmI. En la columna izquierda de esta figura se muestra la lista de dispositivos disponibles en la red. En la parte superior derecha las reglas creadas por un usuario u ordinarias *(ordinary rules)* y en la parte inferior derecha las reglas del sistema o reglas obligatorias *(mandatory rules)*.

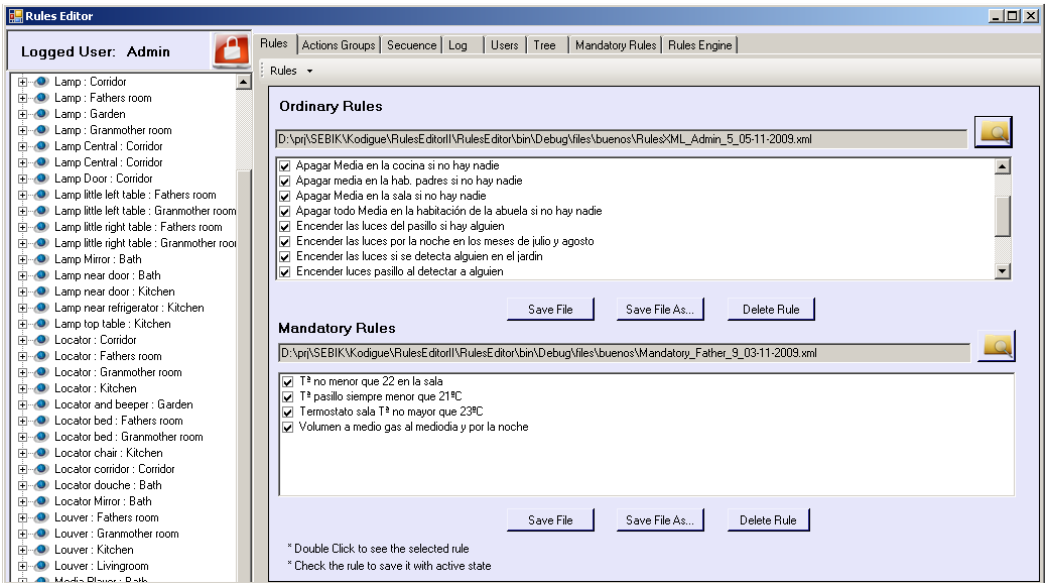


Fig. 6.5 Vista principal del editor de reglas

Después de la creación de las reglas, el editor permitirá al usuario:

a. Guardar reglas: pulsando el botón correspondiente se guardarán las reglas generadas en un fichero con formato XML.

b. Borrar reglas: si en un momento dado se desea eliminar una regla de la lista de reglas, pulsando en el botón correspondiente después de haber seleccionado la regla será eliminada definitivamente.

c. Activación/desactivación de reglas: si por cualquier circunstancia no se desea eliminar una regla, pero tampoco se desea que se cargue en el MR, se puede desactivar. Pinchando en el recuadro al inicio de cada regla, se realizará la acción de activación/desactivación.

En el siguiente apartado, primero, se describirán las funcionalidades generales del editor TAMAmI, y finalmente, se mostrará cómo crear reglas en el editor paso a paso.

6.1.3.1 Gestión de usuarios

Nada más iniciar la aplicación, el editor solicita la introducción del nombre de usuario y contraseña, tal y como se muestra en la figura 6.6.

Fig. 6.6 Pantalla de *login* del editor reglas

El ER dispone de un mecanismo para la gestión de diversos usuarios. La primera vez que se ejecuta la aplicación, se entra como administrador por defecto, y se pueden añadir todos los usuarios deseados. Para la validación, los usuarios escogidos serán los miembros de una familia. A lo largo de este documento se hará referencia a este escenario de experimentación. Los familiares que viven en la casa son: padre, madre, hija, hijo y abuela.

A la hora de la creación de los usuarios, se definirá el rol y el nivel de permisos para cada usuario, como se muestra en la figura 6.7.

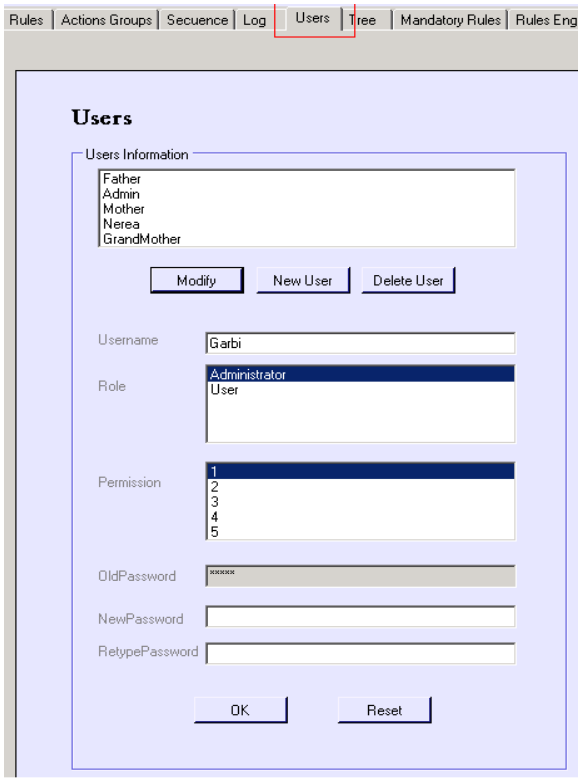


Fig. 6.7 Vista de 'administración de usuarios'

El rol ‘Administrator’ o ‘User’ sirve para dar mayor o menor funcionalidad en el ER a un usuario. Por ejemplo, únicamente un ‘Administrador’ podrá crear/quitar usuarios o modificar los permisos de un usuario. En caso contrario, la pestaña correspondiente del editor estará deshabilitada para ese usuario. De la misma forma, se activarán o desactivarán propiedades en el ER en función del rol del usuario que se ha conectado, como se describirá más adelante.

El permiso define el nivel de prioridad que se le asigna a un usuario. El MR, en caso de detectar un conflicto en la ejecución de las acciones de dos o más reglas simultáneamente definidas por diferentes usuarios, utilizará esta prioridad o nivel de servicio para solventar el conflicto. Cuanto menor sea este valor, mayor prioridad tendrá el usuario. Recalcar que el valor del permiso resultará de vital importancia a la hora de la resolución de conflictos.

6.1.3.2 Edición de reglas

La figura 6.8 muestra la ventana inicial para crear una regla.

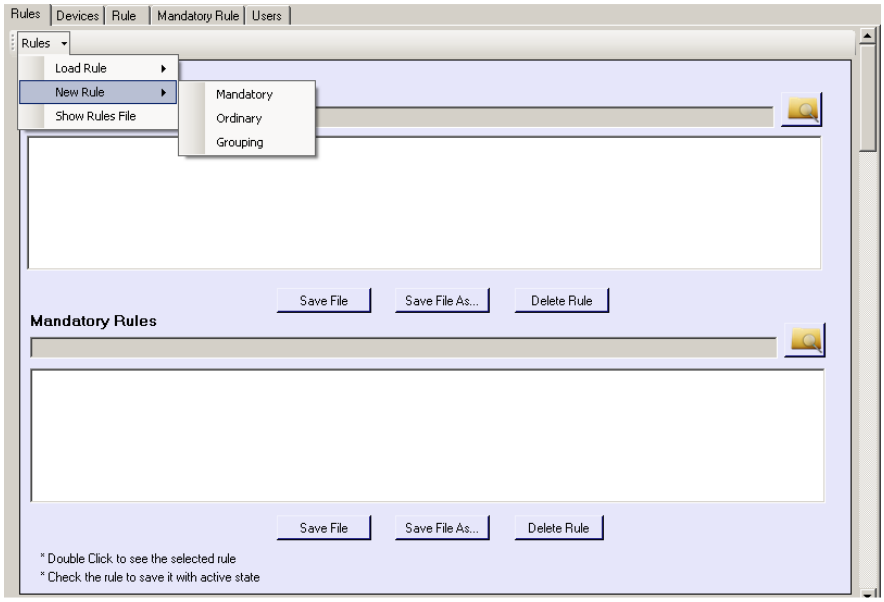


Fig. 6.8 Vista general del menú principal del editor de reglas cuando no existe ninguna regla

6.2 Edición de condiciones

Una vez seleccionado el tipo de regla que se desea generar, se deberá proporcionar información relativa a la condición o las condiciones que la forman. Para ello se deberá seleccionar el tipo de condición que se desea crear (Fig. 6.9). Existen tres tipos de condiciones: simple, temporal, y calculada. Para poder realizar composiciones de condiciones (condiciones compuestas) se utilizarán las operaciones AND y OR (seleccionando ‘New Conjuction’ en el editor).

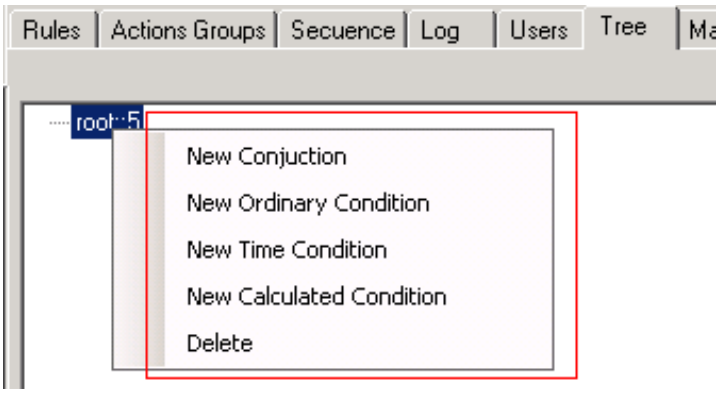


Fig. 6.9 Vista del tipo de condiciones en el ER

6.2.1 Creación de una condición simple u ordinaria

Como ejemplo, se creará una regla para encender las luces de la habitación de la abuela cuando entre por la puerta. La regla sería 'cuando la abuela entre en la habitación, que se encienda la luz'. La condición es 'cuando la abuela entre en la habitación' y esta se muestra en el editor en la parte izquierda de la figura 6.10, también se muestra la creación de la condición. Primero se selecciona el dispositivo, luego el servicio correspondiente, su variable y finalmente la definición y asignación del valor que se desea que tenga la condición. En la parte derecha se muestra la condición resultante que formará parte de la regla.

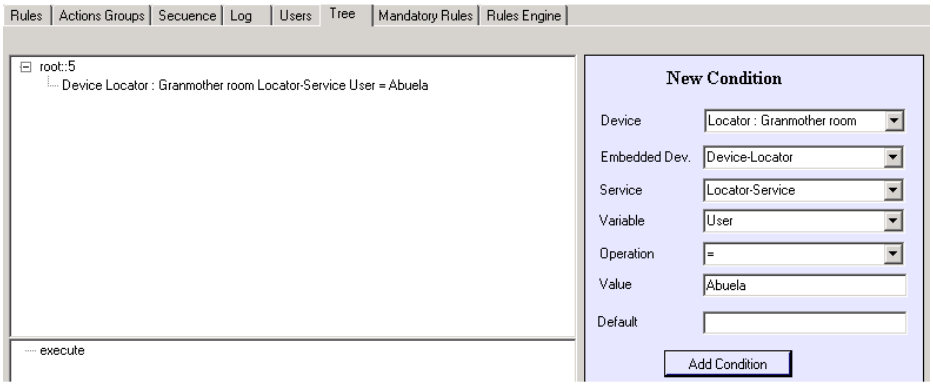


Fig. 6.10 Vista de la creación de una condición simple

6.2.2 Creación de una condición temporal

Por ejemplo, si queremos que durante la hora de la siesta no se pueda poner el volumen de la televisión más alto que la mitad, crearemos una regla cuyas condiciones sean temporales (Fig. 6.11). Imaginemos que la hora de la siesta se considera desde las tres hasta las cuatro y media de la tarde. La regla va a estar compuesta por dos condiciones temporales simples, una de ellas para verificar que son más de las tres de la tarde y otra para verificar que no es más tarde de las cuatro y media de la tarde. Las dos condiciones temporales unidas por el operador lógico AND forman una condición temporal compuesta. La condición formada da la franja horaria que se deseaba acotar.

Si queremos que la regla tenga vigor en todos los días de la semana, entonces habrá que establecer Month = All y Weekday = All.

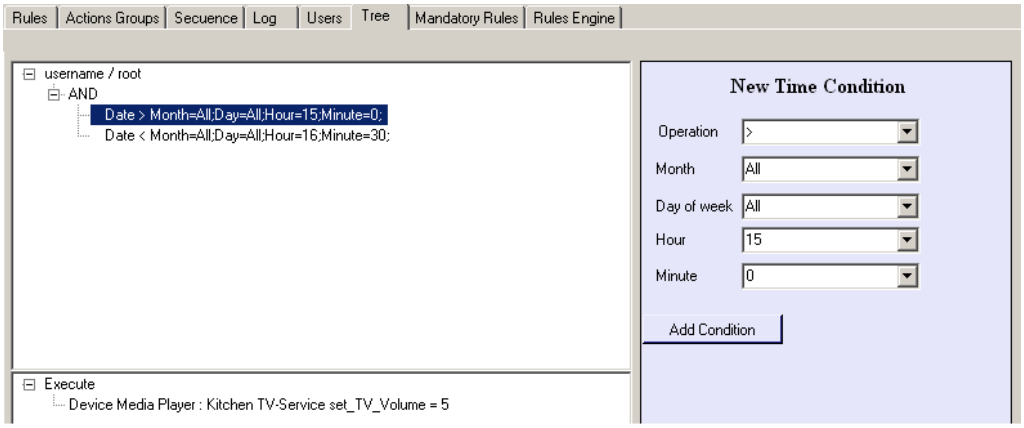


Fig. 6.11 Vista de la creación de una condición temporal

6.2.3 Creación de una condición calculada

Estas condiciones permiten la agrupación y evaluación de variables de dispositivos diferentes.

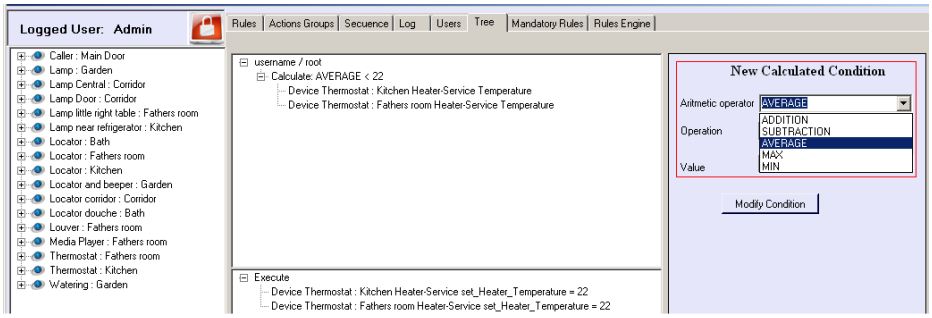


Fig. 6.12 Vista de la creación de una condición calculada

Por ejemplo, en la figura 6.12 se ha generado una condición para calcular la media entre las temperaturas de dos habitaciones, y la condición se cumplirá si el valor es menor que 22. En caso positivo, las acciones serán las de poner la consigna en el termostato de cada habitación a 22ºC.

6.2.4 Creación de una condición sin dispositivo o condición de grupo

Por ejemplo, para crear una regla que apague todas las luces de la sala es necesario definir una o varias condiciones de grupo. Por ejemplo, en la sala puede haber una única lámpara o más de una o se pueden comprar más adelante. Esta condición permite no conocer las lámparas o luces que existen previamente en la sala. Al definir una condición de grupo, no hace falta definir los dispositivos que ofrece el servicio de luz, y que se encuentran en la sala actualmente. El MR en tiempo de ejecución agrupará a posteriori el grupo de dispositivos con servicio de luz, y cuya ubicación es

la sala. En el editor no se especifica el dispositivo o los dispositivos, solamente su servicio, la variable a cambiar y su valor (Fig. 6.13). El MR será el encargado de interpretar y evaluar en cada caso qué dispositivos no conocidos en el momento de la generación de la regla cumplen la condición que se ha descrito. Estas condiciones generalmente forman parte de las reglas de tipo grupo, aunque también se pueden emplear en la creación de reglas ordinarias. Al usar condiciones de grupo, el MR analiza dinámicamente los dispositivos existentes y los convierte en parte activa a la hora de la evaluación de la condición.

Remarcar, que no se selecciona ningún dispositivo en particular. En el momento del descubrimiento de cada dispositivo, esta regla será evaluada, y si se cumplen las condiciones entonces, será realmente añadido como dispositivo.

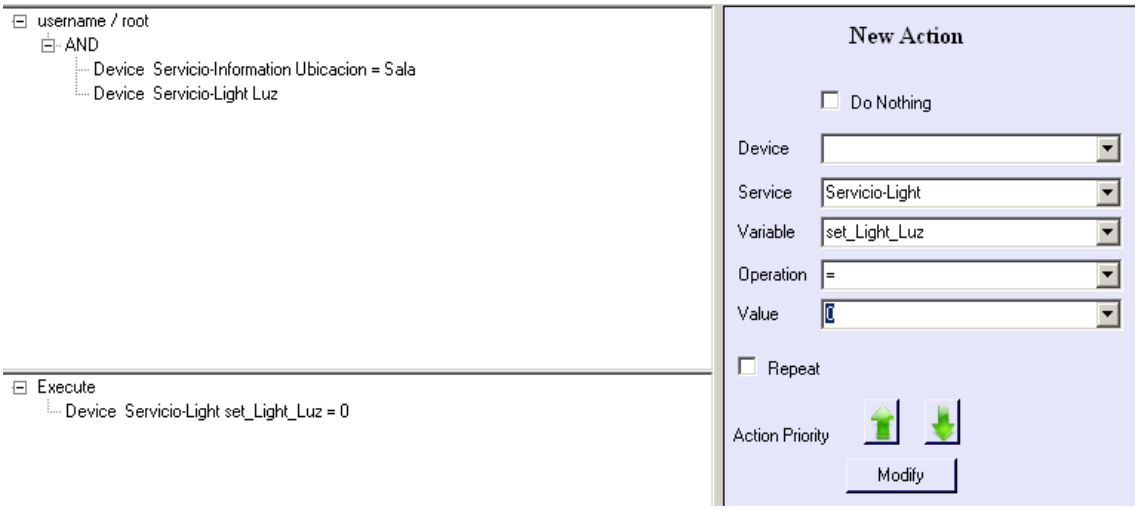


Fig. 6.13 Regla ‘Apagar las luces de la sala’ y su correspondiente condición de grupo

6.2.5 Borrado de una condición

En la fase de creación de la condición, si en algún momento se desea eliminar alguna condición previamente creada, no hay más que colocar el ratón encima de ella y pulsando el botón derecho del ratón se puede eliminar.

6.3 Edición de acciones

En este apartado se explica cómo definir la acción o las acciones a realizar cuando se cumple la condición de la regla. Para ello, primero se seleccionará el tipo de acción a añadir: simple, vinculada (*linked action*), de grupo (*group action*) o calculada (*operation action*) (Fig. 6.14).

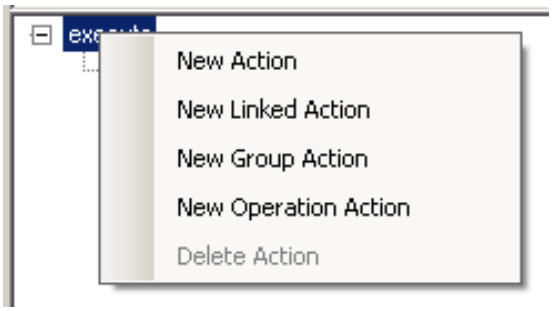


Fig. 6.14 Vista de los tipos de acciones

6.3.1 Creación de acciones simples

Las acciones simples o normales son aquellas definidas por una única acción. El usuario podrá identificar y seleccionar el dispositivo, servicio, variable, y asignar el valor de la acción. Por ejemplo, la regla 'cuando la abuela entre en su habitación, que se enciendan todas las luces' involucra tres acciones simples (Fig. 6.15).

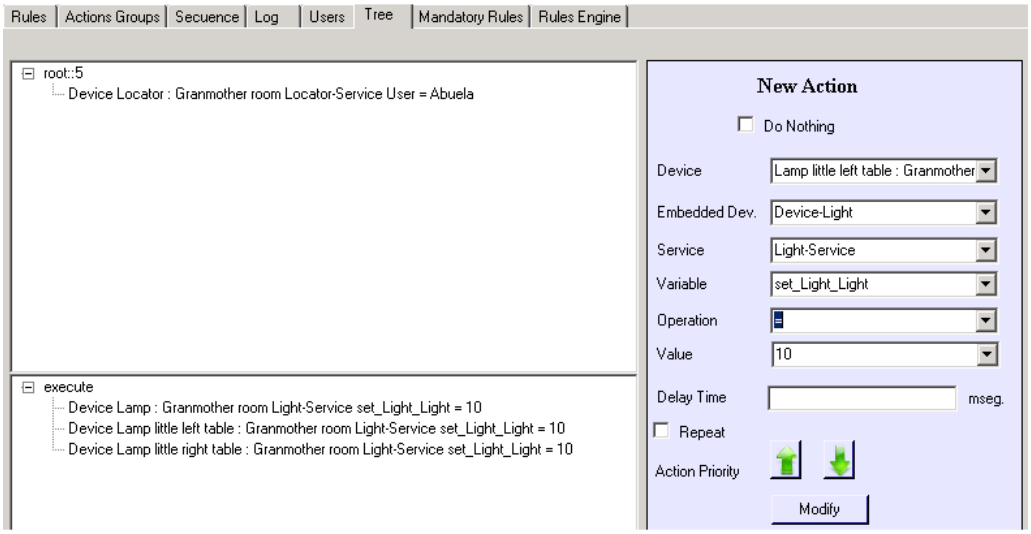


Fig. 6.15 Creación de una acción simple y vista de la regla 'cuando la abuela entre en su habitación, que se enciendan todas las luces'

6.3.2 Creación de acciones vinculadas

La acción de tipo vinculada es la acción que asigna el valor de otra variable en el momento de la ejecución al valor de una acción.

En la figura 6.16 se muestra una regla donde si hay alguien en la cocina, y la televisión está encendida al mediodía o por la noche, el valor del volumen a usar es el que en ese momento tiene la televisión de la habitación de los padres. En la parte derecha de la figura 6.16, se puede observar que el valor se preselecciona de la lista de dispositivos indicando el servicio y variable deseada.

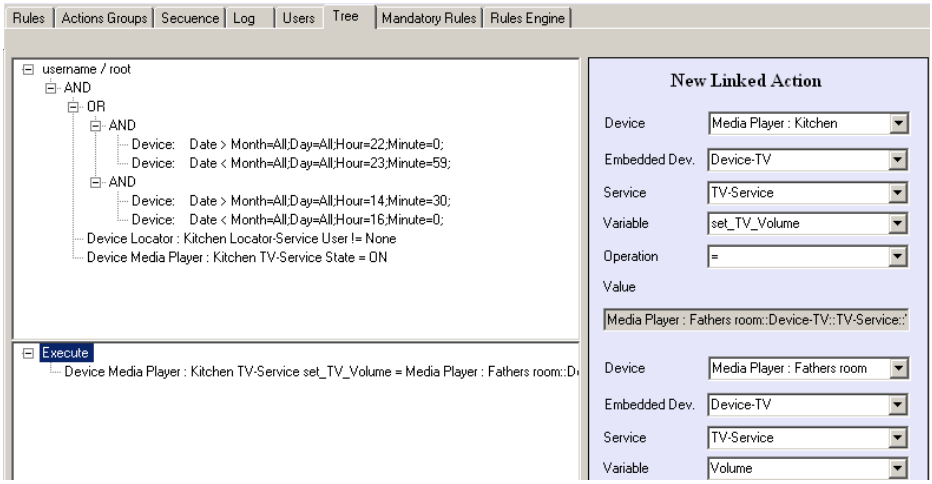


Fig. 6.16 Vista de la creación de una acción de tipo vinculado.

6.3.3 Creación de acciones calculadas

La acción de tipo calculada es aquella acción en la que el valor a asignar se calcula sumando o restando una cantidad al valor de una variable en el momento de la ejecución. De antemano no se conoce el valor que va a tener la acción. Por ejemplo, si la temperatura del baño en algún momento baja de 22 ºC, la acción a realizar es la de subir la temperatura que exista en ese momento en dos grados (Fig. 6.17).

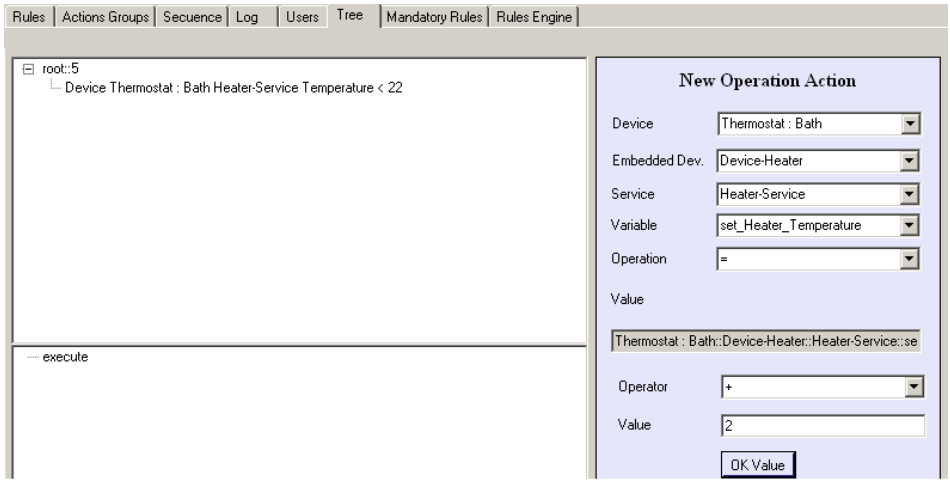


Fig. 6.17 Vista de la creación de una acción de tipo calculada

En la figura 6.17 en la parte superior derecha de la imagen se selecciona la variable de la acción que se desea actuar. En el panel de la acción calculada (parte inferior derecha de la figura 6.17), se selecciona la operación a realizar (suma o resta) y el valor deseado asignando un valor de futuro a la variable.

6.3.4 Creación de acciones de tipo ‘Do Nothing’

Existe una acción especial cuyo uso suele estar asociado con las reglas del sistema, denominado ‘Do Nothing’ o ‘No dejar hacer nada’, y que describe la esencia del concepto de regla del sistema. Por ejemplo, si se desea que nadie pueda modificar el

estado de las luces de la sala entre las dos y las cuatro de la tarde, se creará una regla de tipo ‘Do Nothing’. Para ello, basta crear una acción simple y seleccionar la acción de tipo ‘Do Nothing’ (Fig. 6.18)

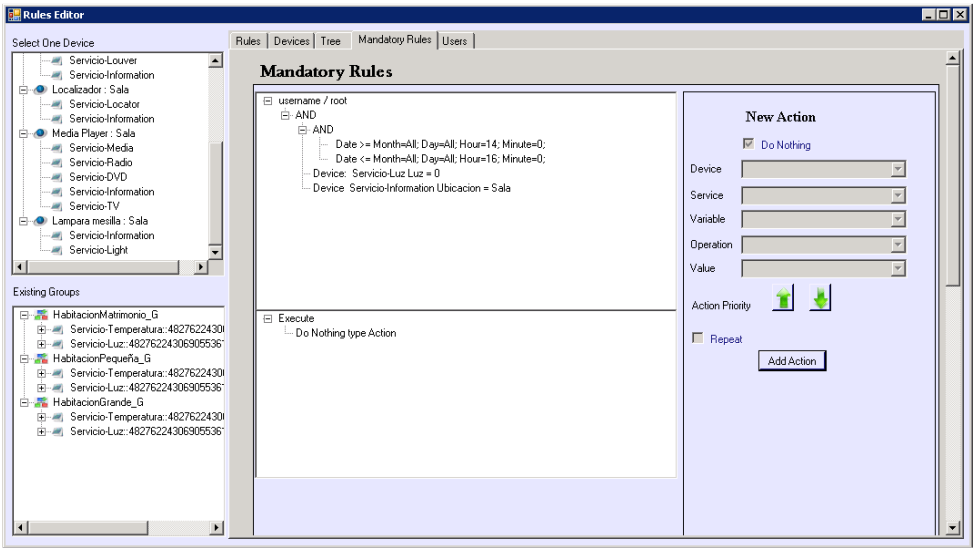


Fig. 6.18 Vista de la regla del sistema de tipo ‘Do Nothing’

Si una regla de un usuario se activa por el MR e intenta modificar el valor de la variable ‘Luz’ cuyo valor es cero (apagada), el MR comprobará los conflictos entre esta regla del usuario y las del sistema. Si el MR encuentra la regla del sistema de tipo ‘Do Nothing’, la acción de la regla del usuario no se ejecutará.

6.3.5 Creación de acciones de tipo grupo

La creación de acciones de tipo grupo implica la realización de varios pasos. Primero se ha de crear la regla de tipo grupo. Por ejemplo, para crear la regla ‘Apagar todas las luces cuando no haya nadie en casa’, debemos crear primero un grupo de dispositivos que ofrece el servicio de ‘luz’ para poder apagarlas. Segundo, crearemos la regla con la condición si ‘no hay nadie en casa’. La peculiaridad de esta regla radica en que de antemano no sabemos cuántos dispositivos de tipo luz hay o puede haber en el futuro, por lo que vamos a necesitar de algún mecanismo para localizarlos y agruparlos. En este caso utilizaremos el mecanismo de crear una regla de tipo grupo.

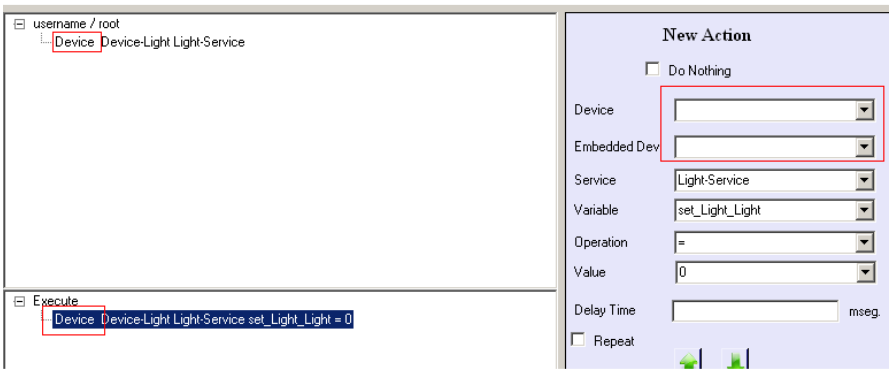


Fig. 6.19 Creación de un grupo de acciones

Como se puede observar en la figura 6.19, al igual que en la creación de la condición de grupo, no se indica ningún dispositivo para la actuación. El MR interpretará esta información, y recopilará y agrupará los dispositivos que cuentan con el servicio de 'luz'. La acción de grupo aparece como si se tratara de una única acción. A esta acción de grupo le llamaremos 'Grupo para apagar todas las de tipo luz'.

Acto seguido, como se puede observar en la figura 6.20, en la parte condicional añadimos todos los dispositivos de localización del hogar, y en la parte de las acciones, utilizamos una acción de tipo grupo, es decir, la que acabamos de crear anteriormente.

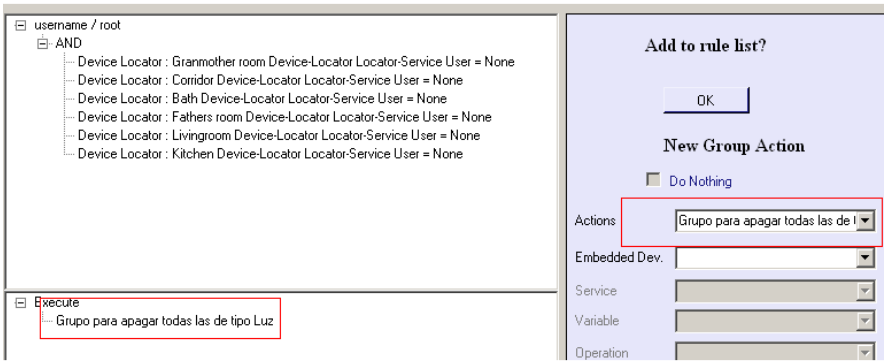


Fig. 6.20 Asignación de un grupo de acciones

De esta manera, sin conocer a priori el número, ni los dispositivos que ofrecen luz, en el momento de la ejecución se apagarán todas las luces. Este mecanismo nos aporta dinamismo en tiempo de ejecución.

6.3.6 Definición de la prioridad de ejecución de una acción y repeticiones

Una regla puede estar compuesta por más de una acción, y el usuario puede asignar el orden de ejecución de las mismas (*Action Priority)* (Fig. 6.21). El parámetro 'Delay Time' o tiempo de espera está muy relacionado con la espera de esa orden de ejecución. Este parámetro establece el tiempo que el MR debe esperar después de la ejecución de la acción definida para ejecutar la siguiente en la lista. Por defecto su valor es cero, es decir, no hay tiempo de espera entre la ejecución de una acción y la siguiente. Se trata de un mecanismo para garantizar que el orden de la ejecución de la lista de acciones puede realizarse incluso cuando el tiempo de respuesta de la ejecución de una acción es lento. Esto es muy útil en la parte de experimentación. Por ejemplo, en un entorno simulado en el que el termostato de una habitación se refresca cada cinco segundos, si se efectúa la acción de puesta de consigna a 22 ºC, hasta pasar cinco segundos no se va a tener respuesta del termostato, por lo que puede interesar esperar a que el termostato reaccione antes de seguir con el resto de tareas de asignación de acciones.

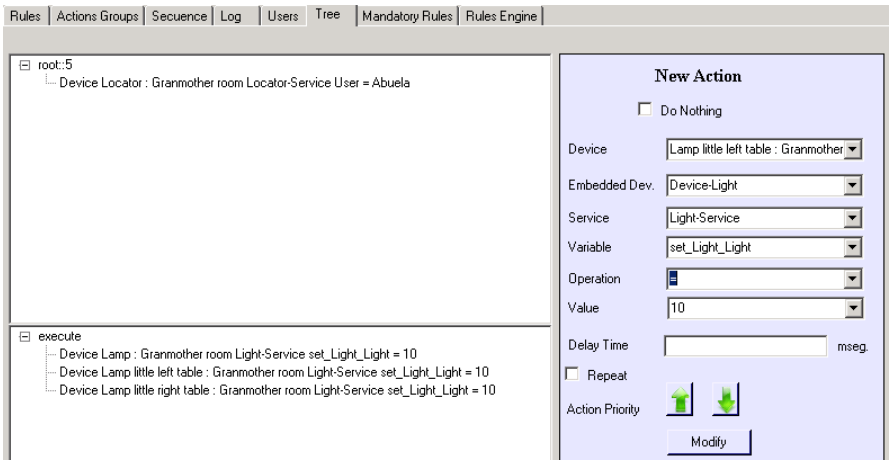


Fig. 6.21 Creación de una acción simple y vista de la regla 'cuando la abuela entre en su habitación, que se enciendan todas las luces'

Además, se puede establecer un modo de repetición de la acción para que dicha acción sea periódicamente ejecutada hasta que el resultado sea satisfactorio. En este último caso, la ejecución de la acción será síncrona; por defecto, las ejecuciones de las acciones son asíncronas. Estos mecanismos serán muy útiles para la validación del MR.

6.3.6.1 Identificación de conflictos

Se han creado varios mecanismos para la resolución de conflictos tanto en el ER como en el MR. En cuanto al ER, la identificación de conflictos con otras reglas existentes permite que el usuario pueda elegir si crear la regla o borrarla ante un conflicto.

Cada uno de los usuarios tiene establecida una prioridad que determina la prioridad de sus reglas en caso de conflicto. Además, un usuario en el momento de guardar una regla puede establecer una prioridad a la regla definida (Fig. 6.22), de tal forma que si a posteriori crea otra regla que pueda estar en conflicto con la anterior, será el valor de la prioridad establecida el utilizado para determinar cuál de ellas es la que prevalece.

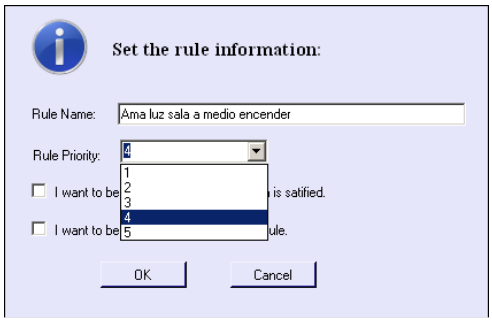


Fig. 6.22 Vista de la asignación de la prioridad a la regla

De todas formas, siempre que se dé una situación de conflicto, el usuario será avisado por el sistema de que dos reglas pueden estar provocando una situación de conflicto.

Cuando un usuario genere varias reglas que pueden producir conflictos (resultados contradictorios sobre las mismas acciones), va a ser notificado en el momento que intente guardar la regla (Fig. 6.23). El ER identifica las acciones coincidentes entre las reglas existentes y la que se desea guardar. El ER pedirá al usuario la confirmación para continuar o para que modifique la última regla editada.

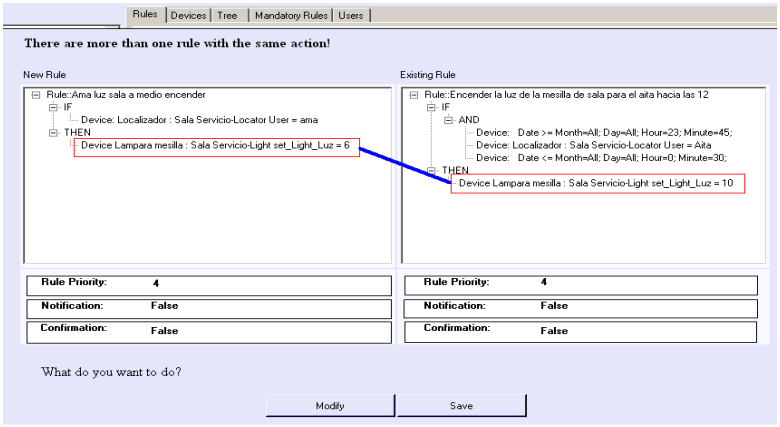


Fig. 6.23 Vista de la información del ER al detectar un conflicto

6.3.6.2 Confirmación y notificación de estado

Dado que en algunas situaciones puede interesar variar el comportamiento automático de las reglas y automatizado de las tareas, el usuario tiene la posibilidad de elegir si desea romper con esta dinámica, y permitir ser interrumpido en sus quehaceres diarios.

Por un lado el usuario puede elegir si desea ser notificado después de la ejecución de las acciones llevadas a cabo por una regla. Y, por otro, si desea realizar una confirmación antes de la ejecución de las acciones de una regla. Ambas opciones podrán ser seleccionadas en el momento de guardar la regla, tal y como se muestra en la figura 6.24.

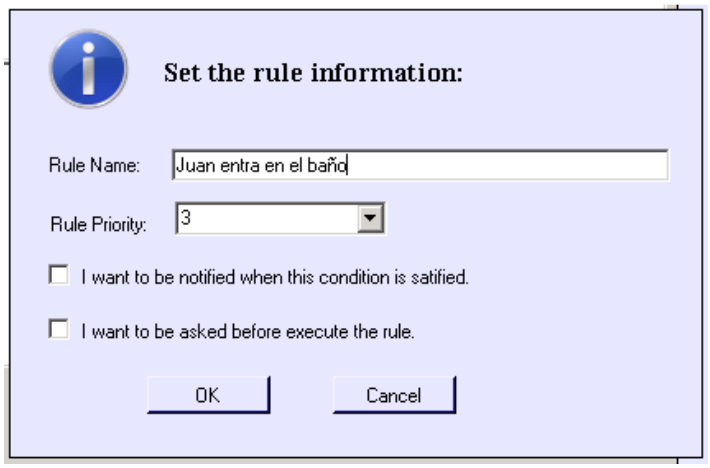


Fig. 6.24 Vista opciones de guardado de la regla

Tal y como se ha mencionado anteriormente, la interacción con el usuario de esta notificación no se presupone que tenga que ser de la forma clásica a través de la pantalla del ordenador, sino que se podría llevar a cabo mediane dispositivos como el móvil, PDA, pantallas de TV, etcétera.

6.3.6.3 Fichero de reglas

Finalmente, las reglas generadas se guardarán en un fichero XML. El fichero de reglas será el utilizado por el MR para cargar las reglas, tanto en local como en remoto. En este último caso, se utilizará la herramienta del ER descrita en el apartado 8, más adelante.

```
<?xml version="1.0" standalone="yes" ?>
- <DocumentElement>
  + <DefaultValues>
  - <Rules>
    + <Rule>
    - <Rule>
        <RuleID>910b5da9-c679-499e-85af-8ea6fbb96ca5</RuleID>
        <RuleName>Al encender la TV se apagan la Radio y DVD</RuleName>
        <RuleType>Device</RuleType>
        <UserName>Admin</UserName>
        <UserPriority>1</UserPriority>
        <Activo>True</Activo>
      - <Conditions>
        - <ComplexCondition>
            <CondUUID>5b1d552e-7484-43d8-876f-f3507ba1c260</CondUUID>
            <Conjunction />
          - <Condition>
              <ComplexConditionUUID />
              <CondName />
              <Type>Device</Type>
              <DeviceID>96aae5b2-1a41-47f5-b2c4-6d89ec1b7639</DeviceID>
              <DeviceName>Media Player : Sala</DeviceName>
              <ServiceID>Servicio-TV</ServiceID>
              <VarName>State</VarName>
              <Operation>=</Operation>
              <Value>ON</Value>
              <DefaultValue />
            </Condition>
          </ComplexCondition>
        </Conditions>
      - <Actions>
        - <ComplexAction>
            <ActionUUID>b1ca9ad0-c77c-48fa-8692-a1137608883e</ActionUUID>
            <Conjunction />
          - <Action>
              <ActUUID>697756560</ActUUID>
              <ActRuleName>set_Radio_State</ActRuleName>
              <Type>Device</Type>
              <GroupActionUUID />
              <DeviceID>96aae5b2-1a41-47f5-b2c4-6d89ec1b7639</DeviceID>
              <DeviceName>Media Player : Sala</DeviceName>
              <ServiceID>Servicio-Radio</ServiceID>
              <ActionName>set_Radio_State</ActionName>
              <Operation>=</Operation>
              <Value>OFF</Value>
              <ActionPriority>2</ActionPriority>
              <Mandatory>false</Mandatory>
            </Action>
          - <Action>
              <ActUUID>166143938</ActUUID>
              <ActRuleName>set_DVD_State</ActRuleName>
              <Type>Device</Type>
              <GroupActionUUID />
              <DeviceID>96aae5b2-1a41-47f5-b2c4-6d89ec1b7639</DeviceID>
              <DeviceName>Media Player : Sala</DeviceName>
              <ServiceID>Servicio-DVD</ServiceID>
              <ActionName>set_DVD_State</ActionName>
              <Operation>=</Operation>
              <Value>OFF</Value>
              <ActionPriority>1</ActionPriority>
              <Mandatory>false</Mandatory>
            </Action>
          </ComplexAction>
```

Fig. 6.25 Vista de un fichero XML de reglas

En la figura 6.25 se puede ver cómo se agrupan los elementos de una regla. Un nodo de tipo regla está compuesto por unos atributos que la describen, quienes a su vez están compuestos por unos nodos hijos que describen las condiciones y las acciones.

El conjunto de nodos junto con su información describen y almacenan toda la información que se han definido en el ER.

6.3.6.4 Carga de las reglas en un motor de reglas determinado

Al ER se le ha añadido una utilidad para cargar las reglas creadas en un MR existente (Fig. 6.26). A través del editor, se muestran los dispositivos que tienen el servicio MR y el editor permitirá cargar un conjunto de reglas en uno de los dispositivos con servicio MR. Se trata de una herramienta para detectar y configurar remotamente los dispositivos que ejecutan un MR. Una vez creadas y guardadas las reglas, se podrán gestionar desde el editor de reglas, cargando y activándolas remotamente en los dispositivos. Se trata de una herramienta que facilita la labor de instalación y mantenimiento de los MRs que podrían estar configurados siguiendo una arquitectura coreografiada en el hogar.

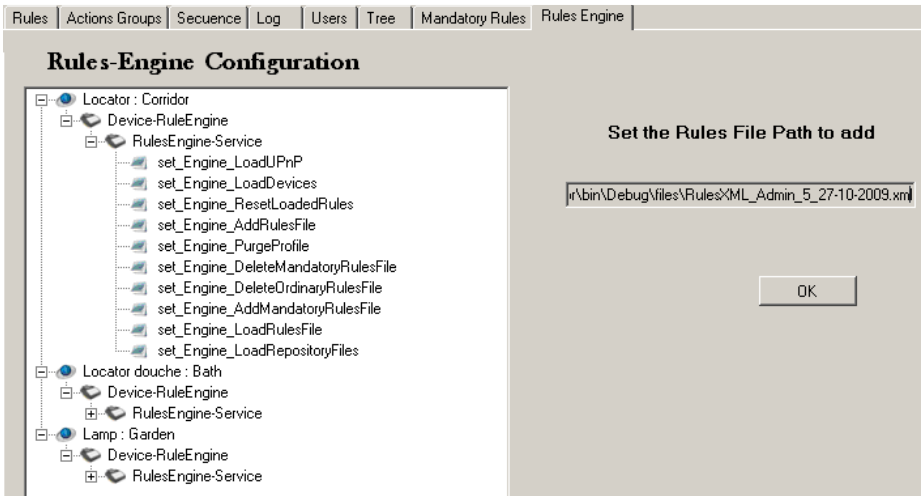


Fig. 6.26 Vista del gestor de los motores de reglas y la carga de una lista de reglas

6.3.6.5 Ejemplos de edición de reglas

A continuación, una vez familiarizados con el entorno de edición de condiciones, acciones y reglas sencillas, se describe un ejemplo más complejo mostrando el proceso de la creación de reglas mediante la definición de un caso de uso. Para más detalle, en el apartado 9.6 del anexo F se describe el proceso de creación de otras reglas.

El proceso de creación de las reglas es muy similar en todos los casos.

- **Regla**: que todas las luces se apaguen cuando no haya nadie en casa. Con esta regla se pretende analizar la creación de una regla de tipo grupo y su inclusión en una regla con condición de tipo servicio ‘luz’.

La condición de la regla es ‘no haya nadie en casa’, mientras que la acción es ‘apagar todas las luces de la casa’. La peculiaridad de esta regla radica en que de antemano no sabemos cuántos dispositivos de tipo localización (para localizar si hay o no alguien en casa) van a existir en la casa, ni cuántos dispositivos de tipo luz hay. En esta situación, se va a crear una condición y una acción de tipo grupo (ambas). Entonces para poder agrupar todos los dispositivos, se ha de utilizar una condición de tipo grupo o sin dispositivo. Para ello no se selecciona ningún dispositivo en la creación de la condición y el MR interpretará esto y agrupará todos los dispositivos que ofrezcan el servicio localización (Fig. 6.27).

Fig. 6.27 Creación de una condición de tipo grupo

En cuanto a la parte de la acción de la regla, ocurre lo mismo, su creación será muy similar a la creación de la condición. A priori, no conocemos los dispositivos de tipo luz existentes que ofrecen el servicio de luz, entonces se agruparán.

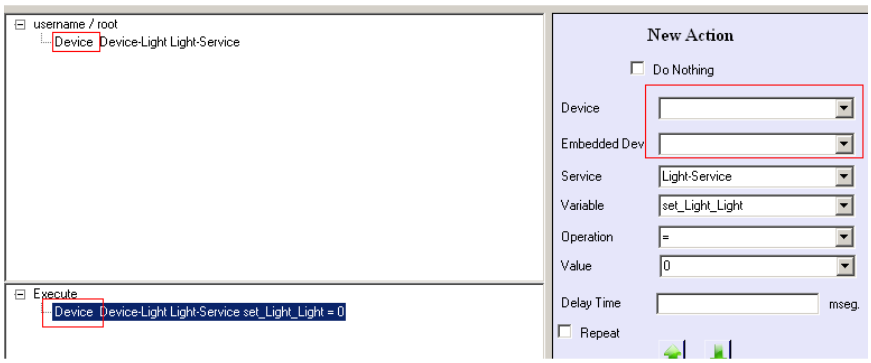


Fig. 6.28 Creación de un grupo de acciones

Como se puede observar en la figura 6.28, al igual que en la creación de la condición de grupo, no se indica ningún dispositivo para la actuación. El MR interpretará esta información, y recopilará y agrupará los dispositivos que cuentan con el servicio de luz.

La acción de grupo aparece como si se tratara de una única acción. A esta acción de grupo la llamaremos ‘Grupo para apagar todas las de tipo luz’.

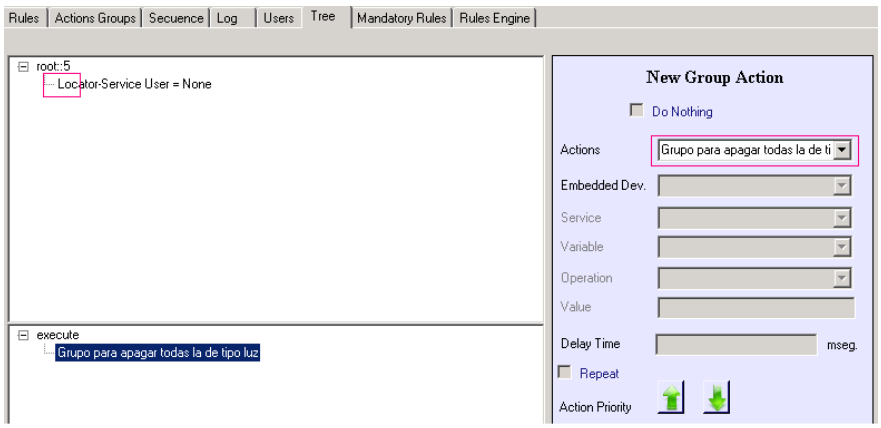


Fig. 6.29 Asignación de un grupo de acciones

Como se puede observar en la figura 6.29, en la parte condicional se crea la condición donde se incluyen todos los dispositivos de localización del hogar, utilizando el mecanismo de condición de grupo, y en la parte de acciones, utilizamos una acción de tipo grupo, es decir, la que acabamos de crear anteriormente. De esta manera, sin conocer a priori el número ni los dispositivos que ofrecen luz, en el momento de la ejecución se apagarán todas las luces gracias a este mecanismo que nos aporta dinamismo en tiempo de ejecución.

6.4 Conclusiones

En este capítulo se han analizado las diferentes necesidades de adaptación y personalización que podrían ser necesarias para un usuario en un entorno como el hogar. Además, para poder personalizar un entorno por un usuario cualquiera, se ha introducido un editor gráfico que permite la creación y edición de las reglas que los diferentes usuarios puedan demandar a hora de la personalización del comportamiento de los elementos que los rodean.

Cabe destacar, que el ER y MR de reglas han sido diseñados para tener en cuenta el dinamismo de los entornos en los que se mueven los usuarios. El MR será el encargado de interpretar y evaluar en cada caso qué dispositivos no conocidos en el momento de la generación de la regla, cumplen con la condición que se ha descrito. Dinámicamente se estarán añadiendo los dispositivos según se descubran convirtiéndolos en parte activa a la hora de la evaluación de las condiciones definidas. La ejecución de dichas reglas por parte del motor de reglas ECA propuesto se ajusta a las necesidades que un entorno del hogar u oficina podría demandar. La flexibilidad que aporta el lenguaje de especificación de reglas unido al editor gráfico permite que los usuarios puedan explotar el sistema de una manera más intuitiva.

En el capítulo siguiente se realizará la validación del sistema propuesto, por una parte analizando el motor de reglas desde el punto de vista de rendimiento y, por otra parte, analizando el *feedback* sobre el editor de reglas de varios participantes. Además, se evaluará una simulación del MR funcionando a lo largo de un día. Los resultados obtenidos, como se verá, han sido totalmente satisfactorios.

No obstante, la optimización de la resolución de conflictos utilizando otras técnicas, o el análisis de otros mecanismos y/o formatos para el intercambio de información entre los diferentes módulos pueden ser aspectos que podrían mejorar el comportamiento del sistema.

Capítulo 7: Experimentación y validación

La validación de las diferentes contribuciones que se han descrito a lo largo de esta tesis, se ha realizado mediante la utilización del propio editor de reglas TAMAmI junto con otras herramientas de validación (por ejemplo, Action Feeder). TAMAmI hace uso de todos los componentes de la plataforma siendo el nexo de unión entre todos ellos. Por un lado, la validación funcional y del comportamiento de todas las contribuciones y, por otro, del rendimiento de todo el middleware, se realizará analizando diferentes aspectos: número de eventos procesados, tiempo de ejecución de un número de reglas elevado o detección de conflictos, entre otros aspectos. Para ello, se han simulado los eventos producidos por los distintos dispositivos embarcados en un hogar inteligente, como pueden ser, lámparas, televisores, actuadores varios, etcétera. A continuación, se describe todo lo referente a la experimentación y validación.

7.1 Introducción

Para llevar a cabo la validación del Motor de Reglas (MR) y del Editor de Reglas (ER) se llevaran a cabo diversas pruebas. El objetivo es medir el rendimiento y el correcto funcionamiento de la plataforma propuesta. En algunos de ellos, se medirá el tiempo de procesamiento y ejecución del MR, mientras que en otros se verificará que la lista de tareas programadas ha sido realizada según lo establecido. Tal y como se describen en [Brebner+04], “la evaluación de un *middleware* debe servir para medir y comparar las prestaciones de un *middleware* específico para diferentes configuraciones”. En este sentido, el banco de pruebas realizado debe servir para evaluar la escalabilidad del *middleware*. Por otro lado, mencionan también que las pruebas realizadas no sólo deben mostrar resultados finales, sino que deben estar acompañadas de toda la información de configuración utilizada para la realización de la prueba. También concluyen que toda información numérica únicamente será aceptable cuando las mediciones estén bien definidas y sean fácilmente reproducibles y repetibles para su verificación. Finalmente, mencionan que un test de medición debe indicar el rendimiento de una aplicación real. En este trabajo se han seguido todas las indicaciones mencionadas para efectuar las pruebas.

A continuación, se describen los cuatro test que se llevarán a cabo. Los tres primeros están relacionados con el MR, mientras que el último lo está con el propio TAMAmI y la experimentación con los usuarios finales.

- *Test 1*: se incrementan sucesivamente el número de reglas hasta 140, y se estudia su rendimiento cuando aumenta el número de reglas linealmente. En este test se tratará de medir la escalabilidad del sistema y la respuesta de la misma con un número diferente de reglas, como se mostrará más adelante. Se ha considerado que 140 reglas es un número suficientemente elevado para

poder personalizar un entorno y se espera que en general no se supere esta cifra.

- *Test 2*: funcionamiento del motor de reglas en un hogar virtual

  En este test se tratará de un hogar o cualquier otra instancia, y estudiar el comportamiento de todos los elementos del *middleware* en dicho entorno. Para un número específico de reglas de entrada se analizará el comportamiento del *middleware* en el período de un día.

- *Test 3*: estudio del funcionamiento del MR cuando los dispositivos aparecen y desaparecen en la red.

  En este test se tratará de constatar que el *middleware* responde según lo esperado frente a la aparición o desaparición de dispositivos a su alcance. A diferencia con el *Test 2*, se tratará de analizar la respuesta del middleware y cómo reacciona frente a fallos de conexión y/o eventos no esperados provenientes de los dispositivos.

- *Test 4*: usabilidad del editor de reglas realizada por usuarios finales.

  Se realizará una encuesta a varios usuarios para recabar información sobre los diferentes aspectos de TAMAmI.

7.2 Simulación de un entorno virtual

Para realizar la validación, se ha creado un entorno virtual que simula un hogar donde distintos dispositivos virtuales están inmersos. La tecnología elegida para la emulación de servicios y dispositivos ha sido UPnP. Estos dispositivos virtuales se han creado a través de un editor propio (Anexo A Dispositivos virtuales UPnP). Básicamente, este editor permite, por una parte, la definición de dispositivos con los servicios y acciones correspondientes, así como la localización de los mismos en una estancia concreta de una casa, oficina, etcétera. Así pues, para la validación se han generado diferentes dispositivos que suelen conformar cada estancia del hogar, por ejemplo, lámparas para la cocina, sala, habitaciones, detectores de presencia para las diferentes estancias, detectores de presión para los sofás de la sala, detectores de presencia en espejos, controladores de humedad y temperatura que emulan un termostato de aire acondicionado, y muchos otros dispositivos embarcados en la casa como se detalla en los siguientes apartados.

A través del editor, y tras definirse todos los servicios y acciones de un dispositivo, el propio editor automáticamente genera el correspondiente dispositivo UPnP. Así se permitirá el intercambio de información entre los distintos dispositivos UPnP. En la parte izquierda de la siguiente figura 7.1 se muestran los dispositivos disponibles en la red del hogar una vez creados con el editor y mostrados por un punto de control genérico. Por ejemplo, *Louver: Kitchen* es el dispositivo actualmente seleccionado y en

la parte inferior derecha se muestran las propiedades de sus variables y funciones de acceso.

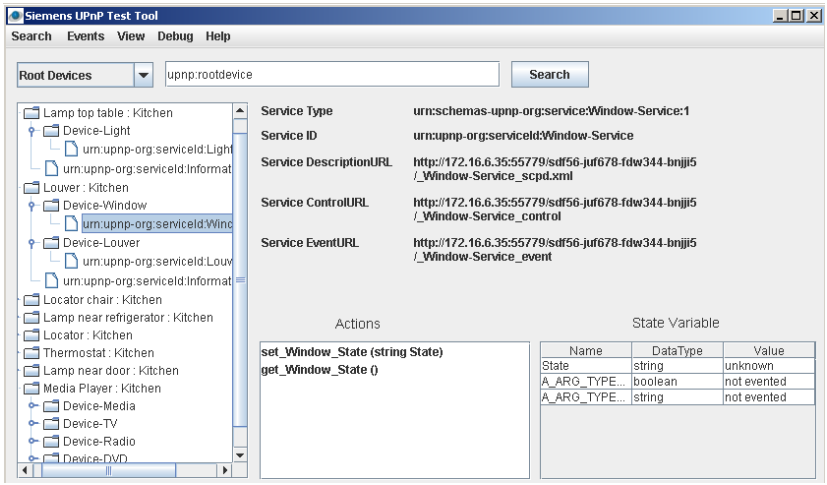


Fig. 7.1 Dispositivos descubiertos por el punto de control genérico de Siemens

Dado que son dispositivos UPnP, a través de un punto de control UPnP o del propio motor de reglas, éstos van a poder ser controlados remotamente.

7.3 Action feeder

Se trata de una utilidad creada para realizar la validación del MR. Para que una regla se valide en el MR, es necesario que se produzcan eventos o, lo que es lo mismo, cambios de variables de los dispositivos. Gracias al *'action feeder'*, se va a poder definir una lista de acciones que provocarán los eventos requeridos. La invocación de una acción implica el desencadenamiento de uno o más eventos, que serán analizados por el motor de reglas. Esta invocación de acciones se corresponde con la secuencia de acciones que un usuario o los miembros de una familia podrían llevar a cabo en su casa o bien con otros cambios de variables que los propios dispositivos pudieran generar de manera automática. Por una parte, el *'action feeder'* permitirá la definición de una lista de acciones, por otra parte, dichas acciones se podrán agrupar y, finalmente, se podrá definir la secuencia de las acciones indicando el tiempo o intervalo de tiempos que ha de transcurrir entre las mismas. El Anexo C describe detalladamente el funcionamiento del *'action feeder'*.

Gracias al 'action feeder', y tras las medidas tomadas al invocar diversas acciones, se observó que se podrían generar un gran número de eventos en el periodo de un segundo, lo que se puede aproximar mucho a la realidad.

7.4 Configuración de los test

En este apartado, se describirán los test y las técnicas para medir el rendimiento del MR. Por una parte, se pretende medir cuál es la capacidad de procesamiento del

motor ante la llegada de un gran número de eventos y la correspondiente evaluación de las reglas, si corresponde.

Lo que vamos a medir es el tiempo transcurrido en la ejecución de cierta cantidad de reglas (esto es, desde 20, 40, 60,…hasta 140 reglas) activadas al mismo tiempo por el cambio producido por una sola variable. Se considera que en un hogar cualquiera la activación simultánea de 140 reglas va a ser una restricción que prácticamente nunca se va a producir. En este sentido es interesante el analizar la respuesta del motor de reglas en particular y el middleware en general, para un número elevado de reglas. Así se valorará el rendimiento del motor en condiciones adversas.

7.4.1 Adaptación al reloj del sistema

Para medir los tiempos de respuesta del MR, a la vista de que las primeras mediciones realizadas con una resolución en segundos no son lo suficientemente exactas, ya que la mayoría de las tareas se ejecutan por debajo de ese tiempo, se ha modificado el modo de cuantificar y medir los eventos producidos dentro del MR. Para ello, dada la baja resolución y precisión del reloj del MS Windows, se ha utilizado el ciclo de reloj del propio procesador para una mayor precisión. En cada acción a medir, se tomará como referencia el tiempo real (ciclo del procesador) de la llegada del evento, y se calculará en cada paso el tiempo transcurrido según los ciclos de reloj transcurridos respecto a la llegada del evento. Es decir, al comienzo de toda medición se almacena el valor numérico en el que se encuentra el procesador, con lo que realizando la resta respecto a los ciclos transcurridos, se obtiene la diferencia de tiempo en unidades menores que el nanosegundo, midiendo el tiempo del procesador transcurrido. El tiempo de referencia almacenado en el momento de la llegada del evento, se transmite a lo largo de la cadena de mensajes de los eventos que se transmiten entre los objetos (variables, condiciones, reglas), de tal forma que en cada momento podemos calcular el tiempo real transcurrido desde la llegada del evento original.

Por otro lado, destacar que cada vez que un evento llega al motor de reglas su procesamiento se ejecuta en un hilo diferente. Cada hilo de ejecución va a tener asignado un identificador que el propio sistema operativo asigna, con lo que el seguimiento de la ejecución de hilos paralelos producidos por eventos diferentes se facilita utilizando el filtrado de los mismos.

Para poder analizar lo que está ocurriendo en periodos de tiempo muy pequeños, se ha diseñado un mecanismo de logado de eventos. Los logs se almacenan en un fichero con formato CSV para poderlos analizar a posteriori en Excel. Es decir, el fichero de log no es más que un fichero con formato CSV con el separador '#', con los campos siguientes: *<Fecha HH:mm:ss:mmm> <NºHiloEjecución> <mensaje1> <mensaje2> <ciclosRelojTranscurridos> <NºCicloRelojReferencia>.* El tiempo de referencia, mencionado anteriormente, es el último campo de cada línea del fichero log (Fig. 7.2). El penúltimo campo es el tiempo transcurrido desde que llegó el evento, y en el resto de información (mensaje1 y mensaje2) describen qué es lo que está ocurriendo en el sistema.

Received_Variable: Ping : Value: from: Locator : Bath # CPU time reference value: # 1980934053288 # 1980934053288
Starting_To_Process variable: Ping Value: of device: Locator : Bath # Elapsed time: # 0,102844914646973 # 1980934053288
Received_Variable: Location : Value: Bath from: Locator : Bath # CPU time reference value: # 1980934422639 # 1980934422639
Starting_To_Process variable: Location Value: Bath of device: Locator : Bath # Elapsed time: # 2,59809556801214E-05 # 1980934422639
Received_Variable: Name : Value: Locator : Bath from: Locator : Bath # CPU time reference value: # 1980934422866 # 1980934422866
Starting_To_Process variable: Name Value: Locator : Bath of device: Locator : Bath # Elapsed time: # 2,17904789575211E-05 # 1980934422866
NOT_SATISFIED_RULE : Grupo para apagar las luces de la habitación de los padres AUTOR: Admin (AND) # Elapsed time: # 0,0362325379342905
Starting_To_Process_Afected_Rule: Grupo para apagar las luces de la sala # Elapsed time: # 1,76000022349209E-05 # 1989818481558
NOT_SATISFIED_RULE : Grupo para apagar las luces de la sala AUTOR: Admin (AND) # Elapsed time: # 0,00155159384782144 # 1989818481558
Starting_To_Process_Afected_Rule: Grupo para encender las luces del pasillo # Elapsed time: # 1,25714301678007E-05 # 1989818932432
SATISFIED_RULE : Grupo para encender las luces del pasillo AUTOR: Admin (AND) # Elapsed time: # 0,0163277735019395 # 1989818932432
Rule: Action Group type. Grupo para encender las luces del pasillo # Elapsed evaluating time: # 0,0163512401715861 # 1989818932432
Starting_To_Process_Afected_Rule: Grupo para apagar las luces # Elapsed time: # 1,14539697084406E-05 # 1989819374763
SATISFIED_RULE : Grupo para apagar las luces AUTOR: Admin # Elapsed time: # 0,00057688896214463 # 1989819374763
Rule: Action Group type. Grupo para apagar las luces # Elapsed evaluating time: # 0,000592812773690511 # 1989819374763
Starting_To_Process_Afected_Rule: Grupo para apagar las luces del baño # Elapsed time: # 1,22920650529606E-05 # 1989819766785

Fig. 7.2 Vista de una parte de un fichero de log

Así vamos a poder visualizar que en un mismo segundo ante la entrada de un único cambio de variable, las diferentes reglas cuyas condiciones incluyen dicha variable se ejecutan en paralelo. El tiempo transcurrido en la evaluación de la condición e invocación de las acciones es menor que un segundo.

7.4.2 Configuración de los ordenadores usados para la validación

El MR se ha ejecutado en un portátil Dell Latitude E5500, con Windows XP Profesional Versión 2002 SP3, a 2,40 GHz y 3,45 GB de RAM. El 'action feeder', por el contrario, se ejecutó en un PC Dell Optiplex 745, con Windows XP Profesional Versión 2002 SP3, Core Duo 2,12 GHz y 2 GB de RAM y todos los dispositivos virtuales en un tercer PC con las mismas características que el anterior. Todos ellos conectados en una red local a 100 Mbps.

7.4.3 Procesamiento de la información

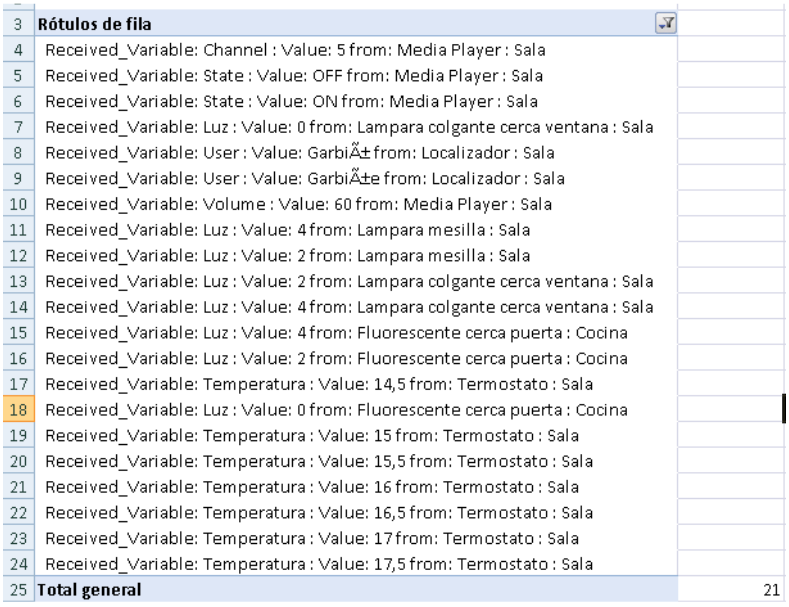
Toda la información relevante de los *logs* se va guardando en ficheros de texto tal y como se ha mencionado anteriormente. Como el acceso al disco puede perturbar y ralentizar la ejecución de procesos en los momentos en los que se pretende coger tiempos de lo que pasa en períodos de tiempo de alrededor del segundo, se ha añadido una opción para deshabilitar la escritura en disco de la información de *logs* hasta la finalización de la prueba.

De esta manera, los *logs* se almacenan en memoria RAM y, una vez finalizada la prueba, el usuario decide volcar la información a disco. Para poder extraer información de los ficheros de *logs* de forma coherente y realizar filtros de búsqueda, agrupaciones,… partiendo del formato CSV utilizado, se ha utilizado una hoja de Excel (Fig. 7.3).

	A	B	C	D	E	F	G
10	8	16:40:22.452	8472	Starting_To_Process_Afected_Rule: Cuando GarbiÃ±e estÃ¡ en la sala en la cocina se apagan	Elapsed time:	7,24E-04	1,64484E+12
11	9	16:40:22.452	8472	SATISFIED_RULE : Cuando GarbiÃ±e estÃ¡ en la sala en la cocina se apagan elementos AUTOF	Elapsed time:	7,67E-04	1,64484E+12
12	10	16:40:22.452	8472	Starting to set action: set_Light_Luz = 0 : Fluorescente cerca puerta : Cocina : Servicio-Light	Elapsed time:	1,42E-03	1,64484E+12
13	11	16:40:22.499	8472	Action executed = set_Light_Luz : Fluorescente cerca puerta : Cocina : Servicio-Light	Elapsed time:	4,97E-02	1,64484E+12
14	12	16:40:22.499	8472	Starting to set action: set_Heater_State = ON : Termostato : Sala : Servicio-Heater	Elapsed time:	4,98E-02	1,64484E+12
15	13	16:40:22.499	8472	Avoiding SET of action value. Value already :ON in :State::Servicio-Heater::Termostato : Sala	Elapsed time:	4,99E-02	1,64484E+12
16	14	16:40:22.499	8472	Starting to set action: set_Heater_Temperatura = 22 : Termostato : Sala : Servicio-Heater	Elapsed time:	4,99E-02	1,64484E+12
17	15	16:40:22.499	8472	Action executed = set_Heater_Temperatura : Termostato : Sala : Servicio-Heater	Elapsed time:	0,057369582	1,64484E+12
18	16	16:40:22.499	8472	Starting to set action: set_Heater_State = OFF : Termostato : Cocina : Servicio-Heater	Elapsed time:	5,75E-02	1,64484E+12
19	17	16:40:22.499	8472	Avoiding SET of action value. Value already :OFF in :State::Servicio-Heater::Termostato : Coc	Elapsed time:	0,057551169	1,64484E+12
20	18	16:40:22.499	8472	Starting_To_Process_Afected_Rule: GarbiÃ±e personaliza sala para ver la tele	Elapsed time:	5,76E-02	1,64484E+12
21	19	16:40:22.499	8472	SATISFIED_RULE : GarbiÃ±e personaliza sala para ver la tele AUTOR: Admin (a2484ae7-51ce-4	Elapsed time:	0,057662357	1,64484E+12
22	20	16:40:22.499	8472	Starting to set action: set_Light_Luz = 0 : Lampara fregadero : Cocina : Servicio-Light	Elapsed time:	5,93E-02	1,64484E+12
23	21	16:40:22.499	8472	Avoiding SET of action value. Value already :0 in :Luz::Servicio-Light::Lampara fregadero : Co	Elapsed time:	0,0593243	1,64484E+12
24	22	16:40:22.499	8472	Starting to set action: set_Light_Luz = 0 : Fluorescente cerca puerta : Cocina : Servicio-Light	Elapsed time:	5,95E-02	1,64484E+12
25	23	16:40:22.515	8472	Action executed = set_Light_Luz : Fluorescente cerca puerta : Cocina : Servicio-Light	Elapsed time:	0,063035665	1,64484E+12
26	24	16:40:22.515	8472	Starting to set action: set_Light_Luz = 0 : Lampara colgante encima mesa : Cocina : Servicio-Li	Elapsed time:	6,33E-02	1,64484E+12
27	25	16:40:22.515	8472	Avoiding SET of action value. Value already :0 in :Luz::Servicio-Light::Lampara colgante encir	Elapsed time:	0,06342957	1,64484E+12
28	26	16:40:22.515	8472	Starting to set action: set_Light_Luz = 0 : Lampara mesilla : Sala : Servicio-Light	Elapsed time:	6,37E-02	1,64484E+12
29	27	16:40:22.515	8472	Avoiding SET of action value. Value already :0 in :Luz::Servicio-Light::Lampara mesilla : Sala	Elapsed time:	0,063726256	1,64484E+12
30	28	16:40:22.515	8472	Starting to set action: set_Light_Luz = 0 : Lampara colgante cerca ventana : Sala : Servicio-Ligl	Elapsed time:	6,39E-02	1,64484E+12
31	29	16:40:22.515	8472	Action executed = set_Light_Luz : Lampara colgante cerca ventana : Sala : Servicio-Light	Elapsed time:	0,07311013	1,64484E+12
32	30	16:40:22.515	8472	Starting to set action: set_Light_Luz = 0 : Lampara colgante cerca puerta : Sala : Servicio-Light	Elapsed time:	7,34E-02	1,64484E+12
33	31	16:40:22.530	8472	Action executed = set_Light_Luz : Lampara colgante cerca puerta : Sala : Servicio-Light	Elapsed time:	0,078176137	1,64484E+12
34	32	16:40:22.530	8472	Starting to set action: set_TV_State = ON : Media Player : Sala : Servicio-TV	Elapsed time:	7,83E-02	1,64484E+12
35	33	16:40:22.530	8472	Action executed = set_TV_State : Media Player : Sala : Servicio-TV	Elapsed time:	0,089886284	1,64484E+12

Fig. 7.3 Vista de una parte del fichero log en hoja de Excel

El fichero log con formato CSV se ha importado a Excel, donde se han generado diferentes filtros para el procesamiento de la información más relevante, por ejemplo, el número de cambios de variable recibidos, el número de reglas satisfechas en un periodo, el número de acciones ejecutadas, etc.



	Rótulos de fila	
3	**Rótulos de fila**	
4	Received_Variable: Channel : Value: 5 from: Media Player : Sala	
5	Received_Variable: State : Value: OFF from: Media Player : Sala	
6	Received_Variable: State : Value: ON from: Media Player : Sala	
7	Received_Variable: Luz : Value: 0 from: Lampara colgante cerca ventana : Sala	
8	Received_Variable: User : Value: GarbiÃ± from: Localizador : Sala	
9	Received_Variable: User : Value: GarbiÃ±e from: Localizador : Sala	
10	Received_Variable: Volume : Value: 60 from: Media Player : Sala	
11	Received_Variable: Luz : Value: 4 from: Lampara mesilla : Sala	
12	Received_Variable: Luz : Value: 2 from: Lampara mesilla : Sala	
13	Received_Variable: Luz : Value: 2 from: Lampara colgante cerca ventana : Sala	
14	Received_Variable: Luz : Value: 4 from: Lampara colgante cerca ventana : Sala	
15	Received_Variable: Luz : Value: 4 from: Fluorescente cerca puerta : Cocina	
16	Received_Variable: Luz : Value: 2 from: Fluorescente cerca puerta : Cocina	
17	Received_Variable: Temperatura : Value: 14,5 from: Termostato : Sala	
18	Received_Variable: Luz : Value: 0 from: Fluorescente cerca puerta : Cocina	
19	Received_Variable: Temperatura : Value: 15 from: Termostato : Sala	
20	Received_Variable: Temperatura : Value: 15,5 from: Termostato : Sala	
21	Received_Variable: Temperatura : Value: 16 from: Termostato : Sala	
22	Received_Variable: Temperatura : Value: 16,5 from: Termostato : Sala	
23	Received_Variable: Temperatura : Value: 17 from: Termostato : Sala	
24	Received_Variable: Temperatura : Value: 17,5 from: Termostato : Sala	
25	**Total general**	21

Fig. 7.4 Filtro del número de eventos recibidos en hoja de Excel.

Los filtros generados se basan en palabras clave que se han utilizado en las descripciones de los eventos que se están trazando. Por ejemplo, en la figura 7.4 se están filtrando los eventos que contengan 'Received_Variable' para contabilizar cuántos cambios de variables han ocurrido en todo el sistema durante la prueba.

En el 'action feeder', comentar que se ha utilizado la opción de llamada asíncrona en la invocación de las acciones (servicios de los dispositivos) a la hora de realizar los cambios de variables en los dispositivos virtuales. Existe la posibilidad de configurar que las llamadas de funciones sean síncronas o asíncronas. Como estamos tratando

de medir el tiempo de respuesta del motor de reglas, se ha optado por utilizar llamadas asíncronas para no perturbar los tiempos obtenidos.

La utilización de llamadas síncronas supone que la ejecución de la lista de acciones simuladas vayan a tardar más en realizarse, ya que se espera hasta recibir la respuesta de confirmación antes de seguir con la simulación del resto de acciones. Como lo que interesa es producir muchos cambios en las acciones de los dispositivos en muy poco tiempo (orden menor del segundo), se utilizarán llamadas asíncronas. Cabe destacar que en una red de área local con todos los dispositivos virtuales y PCs conectados a través de la red local cableada y en el mismo segmento de red, analizando las pruebas realizadas, se puede afirmar que todas las invocaciones de acciones se llevaron a cabo. Aún así, para asegurarse que los mensajes asíncronos llegan a su destino, cada mensaje se envió dos veces.

En cualquier caso, la invocación de acciones asíncronamente o síncronamente únicamente afecta al tiempo de respuesta del MR.

7.4.4 Inicialización del sistema

Para que en cada repetición de las pruebas la simulación de las acciones a través del 'action feeder' provoque el mismo número de eventos que inicialmente hay provoquen el mismo resultado, inicialmente las mismas acciones que forman parte de la secuencia de simulación son invocadas para que los valores de las variables se inicialicen a un valor por defecto. Así, se desencadenarán las ejecuciones de las acciones de las reglas correspondientes.

A continuación se describirán cada uno de los tests que se llevaron a cabo y los resultados obtenidos.

7.5 Test 1: Aumento lineal del número de reglas en el motor de reglas

En este apartado se describirá la creación de las reglas del Test 1, y después se mostrarán los resultados obtenidos.

7.5.1 Creación de reglas

Primero, se han creado las 140 reglas diferentes con la ayuda de TAMAmI que se activan ante un mismo evento y con la misma condición. En la figura 7.5 se muestra un ejemplo de las reglas generadas, donde en la parte superior se indica la condición que ha de cumplirse para que la acción (parte inferior) se ejecute. La regla es 'cuando el termostato del baño se ponga en marcha, se encienda la radio de la sala'.

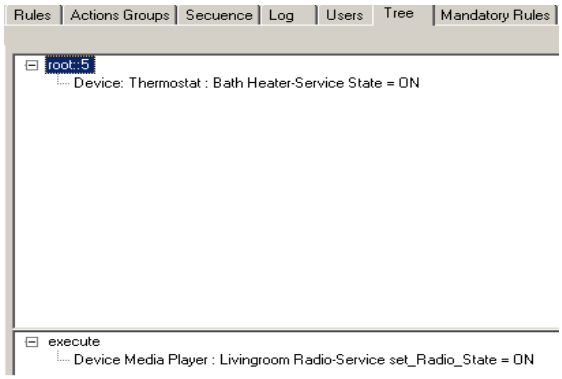


Fig. 7.5 Vista de una de las 140 reglas definidas por el editor de reglas

El evento desencadenante para los test ha sido que el 'termostato del baño se ponga en marcha', es decir, el MR recibirá un evento asociado con la variable lógica del termostato y cuyo valor es ON. A partir de ahí, el MR evaluará si se satisface la condición de cada regla, y si es así, ejecutará la acción correspondiente.

Cara a facilitar la carga de reglas, las 140 reglas se han agrupado en ficheros XML de reglas conteniendo 20 reglas. En la figura 7.6 se muestran 20 reglas y cómo están guardadas en un único fichero.

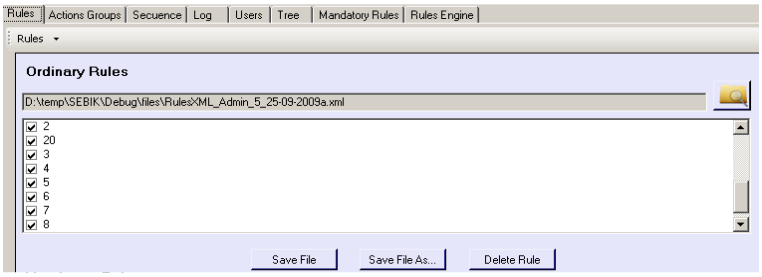


Fig. 7.6 Imagen del contenido de un fichero de reglas en el editor de reglas.

El nombre de las reglas existentes es 2, 20, 3, 4, y así sucesivamente. Después de definir todas las reglas necesarias, se irán cargando en el MR mediante el uso de ficheros XML (con el formato y contenido explicado en el apartado 6.3.6.3). La figura 7.7 muestra las primeras 20 reglas cargadas y activas en la parte izquierda de la figura. Se puede observar también que todas ellas tienen la misma condición para que reaccionen simultáneamente al cambio de la misma variable.

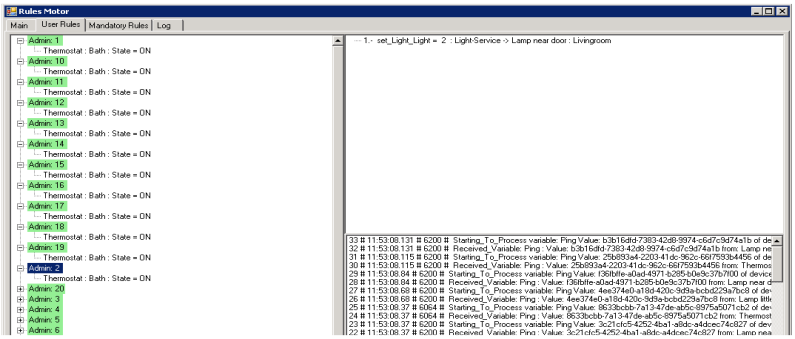


Fig. 7.7 Reglas cargadas en el motor de reglas

A continuación, analizaremos los datos y veremos la eficiencia del MR y si el tiempo transcurrido en el procesamiento es dependiente del número de reglas utilizado.

7.5.2 Ejecución del Test 1

En cuanto a la ejecución del Test 1, se ha repetido 12 veces dicho test con cada número diferente de reglas para comprobar que los resultados no divergían. De esta forma, se pretende que las tareas propias del sistema operativo no condicionen (tal y como se mencionaba en la introducción) los resultados finales, siendo el valor promediado la media de los tiempos de ejecución de los 12 test para cada grupo concreto de reglas. En total, por lo tanto, se han realizado 12 x 7 test (grupos de 20, 40, 60, 80, 100, 120 y 140 reglas), un total de 84 evaluaciones.

7.5.3 Resultados del Test 1

Los datos obtenidos han sido los que se muestran en la tabla 7.1, donde cada columna muestra los valores obtenidos en las sucesivas repeticiones de las pruebas para el número de reglas correspondiente.

Número de reglas	20	40	60	80	100	120	140
Prueba 1	0,251	0,330	0,227	0,537	0,636	0,518	0,860
Prueba 2	0,129	0,264	0,976	0,455	0,461	1,263	0,853
Prueba 3	0,158	0,164	0,281	0,404	0,480	1,039	0,665
Prueba 4	0,156	0,354	0,515	0,446	0,553	0,485	0,718
Prueba 5	0,162	0,110	0,187	0,366	1,123	0,869	0,587
Prueba 6	0,134	0,167	0,476	0,558	1,159	1,351	1,101
Prueba 7	0,285	0,274	0,501	0,404	0,336	0,648	1,769
Prueba 8	0,229	0,266	0,424	0,502	0,374	1,184	0,760
Prueba 9	0,066	0,284	0,359	0,390	0,677	1,252	1,110
Prueba 10	0,082	0,295	0,398	0,265	0,461	0,642	0,627
Prueba 11	0,055	0,279	0,205	0,510	0,544	0,485	0,882
Prueba 12	0,149	0,212	0,474	0,365	0,552	0,600	0,816
Tiempo medio en sg.	0,155	0,250	0,419	0,435	0,605	0,872	0,898

Tabla 7.1 Resultados de los test

Al dibujar los datos, obtenemos la gráfica que se muestra en la figura 7.8. En ella se puede observar por ejemplo, que en el caso más desfavorable para el motor de reglas (procesamiento de 140 reglas que se activan simultáneamente a la llegada del evento, termostato baño=ON), las evaluaciones de las condiciones que se ejecutan en paralelo en hilos diferentes y la ejecución de las 140 acciones correspondientes por cada regla activada tarda como media aproximadamente 0,9 segundos. Por otra parte, se puede observar que el tiempo de respuesta sigue un comportamiento o tendencia casi lineal ante un aumento del número de reglas.

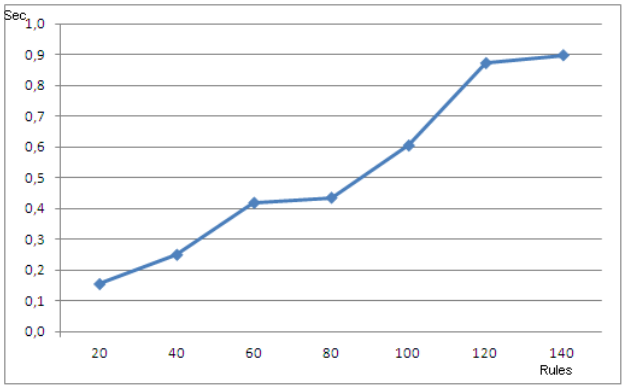


Fig. 7.8 Respuesta en segundos del motor de reglas por cada grupo de reglas

7.6 Test 2: Funcionamiento del motor de reglas en un hogar virtual

En la sección anterior, se ha validado el MR ante el suceso de un evento que desemboca en la validación de un número elevado de reglas. A continuación, vamos a describir el test realizado para evaluar el sistema en un hogar virtual, aproximándonos más a la realidad. En primer lugar, se describirá el número de dispositivos que existe en un hogar virtual (apartado 7.6.1) y las reglas existentes en dicho hogar (apartado 7.6.2). Por otro lado, será revisado el proceso de cargar dichas reglas en el motor de reglas. Finalmente, se simularán las diferentes acciones (eventos) que puedan ocurrir en un hogar. Por ejemplo, se simularán los movimientos de un usuario en el hogar y los eventos que producen los distintos elementos del hogar a través del 'action feeder'. Esta simulación de acciones provocará que el MR actúe según los mismos. Al final se mostrarán los diferentes valores del número de eventos producidos en el sistema demostrando que el sistema reacciona y funciona de manera correcta y en un tiempo de respuesta aceptable.

7.6.1 Testbed o campo de pruebas

En cuanto a la distribución de los diferentes elementos o dispositivos en el hogar, señalar que para la evaluación se han utilizado siete PCs conectados en una red local cableada. En la siguiente figura se muestra el despliegue de los PCs utilizados.

Cada PC simulará el funcionamiento de uno o varios dispositivos (sensores, actuadores, TV, electrodomésticos) en una estancia del hogar. Por ejemplo, un PC simulará los dispositivos existentes en el salón, otro en la sala, y así sucesivamente. Al final, para la validación se contó con un total de 55 dispositivos distribuidos como a continuación se detalla. Uno de los puntos que se trató de abordar fue el de emular no sólo el estado natural de los dispositivos, sino también las condiciones menos favorables como son las posibles sobrecargas del número de eventos procesados por el motor de reglas.

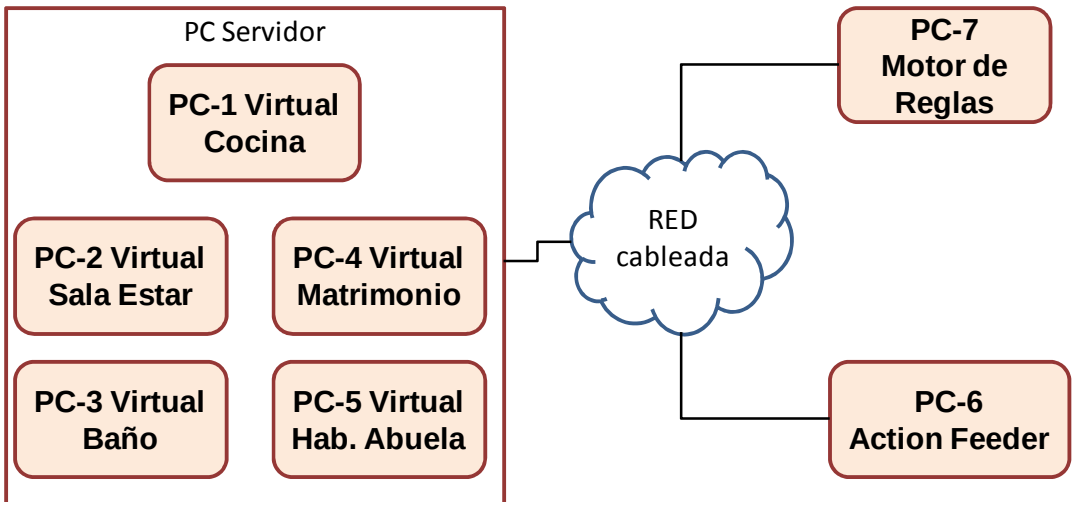


Fig. 7.9 Diagrama de red de los PCs utilizados

Para facilitar la validación del MR, se utilizaron cinco máquinas virtuales sobre un mismo PC Servidor y cada máquina virtual albergó los dispositivos virtuales de cada estancia de la casa.

La herramienta de virtualización utilizada ha sido WMware para Windows [VMware10]. Por otro lado, mediante una pequeña aplicación que se ha desarrollado, en el inicio de sesión automática de cada máquina virtual, se ejecuta la carga de los dispositivos para cada máquina. A continuación de describen los detalles de configuración para cada PC (virtual o físico) utilizado.

7.6.1.1 PC-1 virtual: Cocina

Los dispositivos albergados en nuestra cocina son los siguientes:

- Lámpara encima de la mesa.
- Lámpara cerca del frigorífico.
- Lámpara cerca de la puerta.
- Control de ventana y persiana.
- Localizador.
- Termostato.
- Multimedia: TV, DVD y radio.

7.6.1.2 PC-2 virtual: Sala de estar

Los dispositivos albergados en nuestra sala de estar son los siguientes:

- Lámpara encima de la mesilla.
- Lámpara cerca de la ventana.
- Lámpara cerca de la puerta.
- Control de ventana y persiana.
- Localizador puerta.
- Localizador sofá.
- Termostato.

- Multimedia: TV, DVD y radio.

7.6.1.3 PC-3 virtual: Baño

- Lámpara del espejo.
- Lámpara del techo.
- Control ventana.
- Localizador entrada.
- Localizador espejo.
- Multimedia: radio.
- Localizador ducha.
- Termostato.

7.6.1.4 PC-4 virtual: Habitación matrimonio

- Termostato.
- Multimedia: TV y radio.
- Luz techo.
- Luz mesilla derecha.
- Luz mesilla izquierda.
- Localizador cama.
- Localizador puerta.
- Control de ventana y persiana.

7.6.1.5 PC-5 virtual: Habitación abuela

- Termostato.
- Multimedia: TV y radio.
- Luz techo.
- Luz mesilla derecha.
- Luz mesilla izquierda.
- Localizador cama.
- Localizador puerta.
- Control de ventana y persiana.
- Localizador puerta.

7.6.1.6 PC-6: Action feeder

En el PC 6 donde normalmente se ejecuta el editor de reglas, también se va a ejecutar el ‘action feeder’ y varios dispositivos de carácter genérico, por lo que la distribución de elementos que se ejecutan en este PC queda como sigue:

- ‘action feeder’.
- Dispositivos del exterior o del jardín.
    - Localizador puerta exterior.
    - Portero telefónico.
    - Riego.

    - Luz puerta entrada.
    - Luz jardín.
    - Temperatura exterior (calle).
- Dispositivos varios del interior.
    - Luz pasillo.
    - Termostato pasillo.
    - Localizador pasillo.
    - Luz escaleras.

7.6.1.7 PC-7: Motor de Reglas

En un portátil únicamente se va a ejecutar el motor de reglas. De esta manera se pretende que la CPU esté dedicada únicamente a las tareas del motor de reglas para obtener unas mediciones y tiempos de respuesta que no se vean afectados por otras tareas que no sean más que las necesarias del sistema operativo. En cuanto al portátil, se trata de un Dell Latitude E5500, con Windows XP Profesional Versión 2002 SP3, 2.39 GHz y 3,45 GB de RAM.

7.6.2 Descripción de las reglas

Tras describir el hogar virtual junto con los dispositivos que lo forman, en esta sección se describen algunas de las 66 reglas que se han creado.

En cuanto a las personas que viven en esta casa y sus prioridades asignadas son:

1. Padre (Juan): prioridad 1
2. Madre (María): prioridad 1
3. Abuela (Juanita): prioridad 2
4. Hija (Nerea): prioridad 3

En primer lugar, se describirán las reglas de este hogar. Estas reglas servirán para entender la simulación de las acciones que tendrán lugar. Esto es, la descripción de las reglas ayudará a entender mejor la simulación de las acciones en dicho hogar dada la relación existente entre ambas para llevar a cabo la validación.

Primero, olvidándonos de las personas del hogar, describiremos el tipo de funcionamiento que tendrá el hogar desde el punto de vista del sistema, y en los apartados posteriores se describirán las reglas de cada uno de los usuarios. Como se verá posteriormente se ha tratado de elegir un conjunto de reglas que servirán para demostrar cómo se pueden reflejar diferentes necesidades de los usuarios de un hogar u oficina. Por otro lado, se ilustrará también cómo se resuelven los conflictos entre reglas (descritas en la apartado 5.4.1.1) de diferentes usuarios con jerarquías o prioridades diferentes.

7.6.2.1 Reglas genéricas o del sistemas

A continuación, se describen todas las reglas del sistema que son:

1. Todas las luces apagadas cuando nadie se encuentre en la casa.

2. Todas las televisiones, radios y DVDs deben apagarse cuando nadie se encuentra en la habitación correspondiente.
3. A partir de las 11 de la noche y hasta las 7 de la mañana (por la noche) no se puede poner el volumen de los dispositivos de audio de la sala más alto que la mitad de lo posible. Entre las 3 y las 4 de la tarde (hora de la siesta) no se podrá poner el volumen de los dispositivos de la sala más alto que la mitad de lo posible.
4. Nadie podrá poner una temperatura mayor que 23 ºC en el termostato de la sala.
5. Nadie podrá poner una temperatura menor que 22 ºC en el termostato de la sala en los meses de junio-julio-agosto.
6. En el pasillo no se podrá poner una temperatura superior a 21ºC.
7. En los meses de julio y agosto se encenderán las luces del jardín desde las 10 de la noche hasta las 12 y media.
8. Si se detecta a alguien en el jardín por la noche que se enciendan las luces.
9. Entre las 5 y las 5:20 de la mañana se pondrán en marcha los aspersores (julio y agosto).
10. A las 5:20 de la mañana se pararán los aspersores.
11. Si entra alguien en el jardín se paran los aspersores.
12. Si alguien es detectado en el pasillo, se encienden las luces.

A continuación, describiremos las preferencias de comportamiento para cada persona de la casa.

7.6.2.2 Reglas del padre: Juan

1. Entre las 11 de la noche y las 8 de la mañana nadie podrá ver el canal 69 de la TV.
2. Cuando entre al jardín desde la calle, que el sistema diga “Hola Juan. ¿Todo controlado?”. El detector de localización de la puerta del jardín incorpora la función de locutor de texto (Text-To-Speech) (Véase Anexo A para más detalle).
3. Si entre las 3 de la mañana y las 4 de la tarde me siento en el sofá de la sala, se pone en marcha el canal 2 en la TV de la sala.
4. Siempre que entre en la habitación de matrimonio, que se encienda la luz de la mesilla de noche (se supone que prácticamente de día no entro en la habitación).
5. Siempre que entre en el baño se encienden las luces del espejo y la general.

6. Por la mañana entre las 10 y 10 y media, si no hay nadie en las habitaciones y baño, que se abran las ventanas.
7. A las siete de la mañana se pone en marcha la radio en el canal 2 y volumen bajito.
8. Cuando salga de la habitación, que se apague la radio.
9. Si me coloco delante del espejo, me pide confirmación antes de encender la luz (para afeitarse o arreglarse).

7.6.2.3 Reglas de la madre: María

1. Si la temperatura media en la sala, cocina y pasillo es menor que 20ºC poner la consigna de la temperatura a 24ºC en los termostatos.
2. Si estoy en la cocina se pone el canal 5 en la TV, por la mañana.
3. Si estoy en la cocina se pone el canal 6 en la TV, por la tarde.
4. Si entro en mi habitación se enciende la luz de mi mesilla y se baja la persiana si es de noche.
5. Si me coloco delante del espejo se enciende su luz.
6. Si estoy en la ducha se apaga la luz del espejo.

7.6.2.4 Reglas de la abuela: Juanita

1. En la sala desde las 16:00 a las 17:30 quiero ver el canal 7.
2. Si me siento en el sofá, se bajan las persianas y se enciende la luz de la mesilla.
3. Cuando entro en mi habitación y baño, se encienden todas las luces.
4. Cuando salga de la habitación y baño, que se apaguen las luces.
5. Cuando me meto en la cama, que se abra la ventana y se cierre la persiana.
6. A las 7:00 se cierra la ventana de la habitación.
7. Si son más de las 8:00 y me levanto de la cama se abre la persiana.
8. A las 7:30 se pone en marcha la radio en el canal 1.
9. Cuando salga de la habitación, que se apague la radio.
10. Si me coloco delante del espejo, que se encienda su luz.
11. Si estoy en la habitación de noche, que se enciendan las luces de las mesillas de noche.
12. Si me meto en la cama, que se apaguen las luces de las mesillas de noche.

7.6.2.5 Reglas de la hija: Nerea

1. Cuando estoy en la sala, quiero una temperatura de 24 ºC y que se encienda la TV en el canal 8.
2. Si entro en el baño, que se cierre la ventana y se enciendan todas las luces
3. Por la noche si estoy en el pasillo, que se encienden las luces al mínimo.
4. Si entro a la cocina, que se encienda la TV en el canal 10.

7.6.3 Simulación de las acciones

En primer lugar, se ha descrito el número de dispositivos que existen en el hogar virtual (apartado 7.6.1), las reglas existentes en el hogar (apartado 7.6.2) y a continuación se describe cómo se ha llevado a cabo la creación de las 79 acciones que desencadenarán la evaluación de las reglas definidas.

Además, se definirán los valores para inicializar el sistema, de manera que partamos de un estado inicial en el hogar virtual al que le vamos a aplicar cambios (simular las acciones) y que monitorizaremos. De esta manera, cada vez que repitamos una simulación partiremos de la misma información de partida. Mediante el 'action feeder' se han creado las acciones que queremos que ocurran en el sistema, en el 'Anexo C 'action feeder'' se detalla el proceso de creación de dichas acciones.

Por una parte, como se ha comentado, se han creado acciones para inicializar las variables de los dispositivos. Por otra parte, se han creado acciones que simulan los movimientos de las personas que viven en el hogar, cambios de temperatura, etc.

Por ejemplo, si queremos que la regla de Juan 'Cuando salga de la habitación, que se apague la radio' se evalúe, se creará la acción 'Juan sale de la habitación' que se utilizará en la parte de la evaluación. Para poder simular la ejecución de la acción de 'apagar la radio', por otro lado se creará también la acción que realiza la simulación de 'apagar la radio'.

7.6.3.1 Reloj interno personalizable

En la figura 7.10 se muestran las opciones de configuración del reloj, donde si optamos por personalizar el reloj, podremos elegir la nueva hora de referencia y la frecuencia de la misma. En el 'Anexo B Motor de reglas' en el apartado 7.7 se detallan las características del reloj interno.

Resulta imprescindible la sincronización de los relojes lógicos del *'action feeder'* y el motor de reglas.

Una vez sincronizadas las acciones que se configuran para ejecutarse a una hora concreta, nos aseguramos que el motor de reglas también los procesa en la hora estipulada, convirtiendo el tiempo del test (toda la simulación) en un fiel reflejo de la realidad, como si se hubiera realizado en tiempo real. Es aquí donde también se permite realizar un test de 24 horas en 24 minutos.

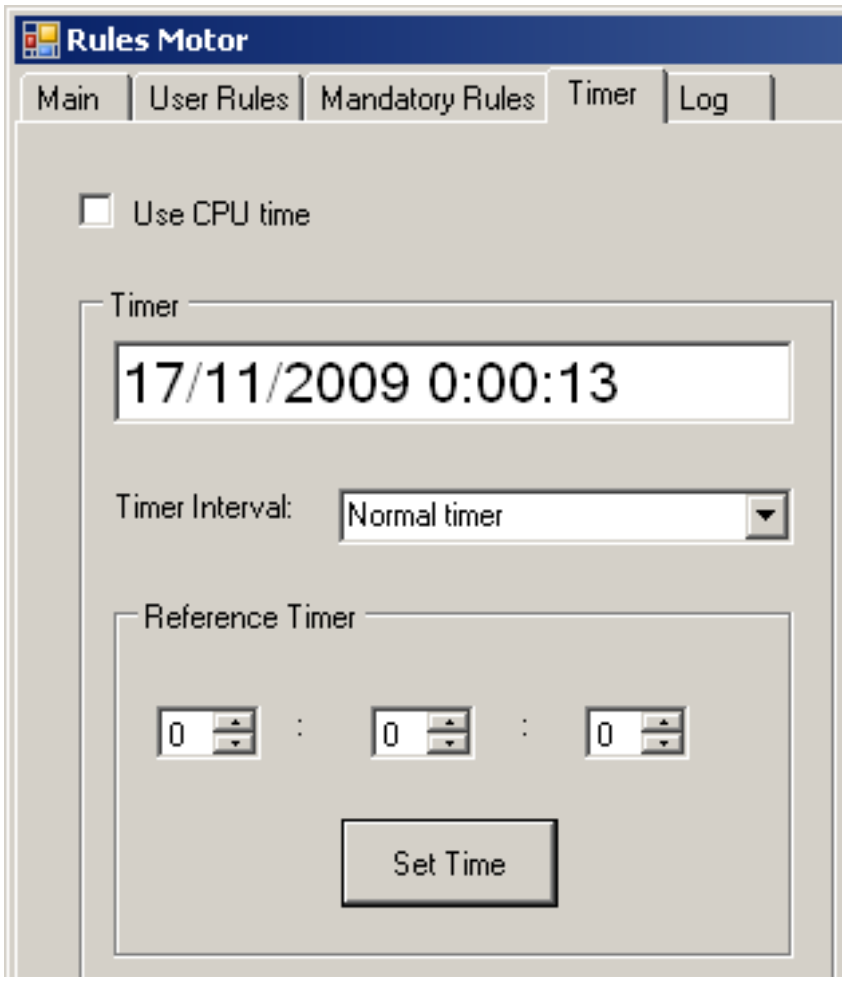


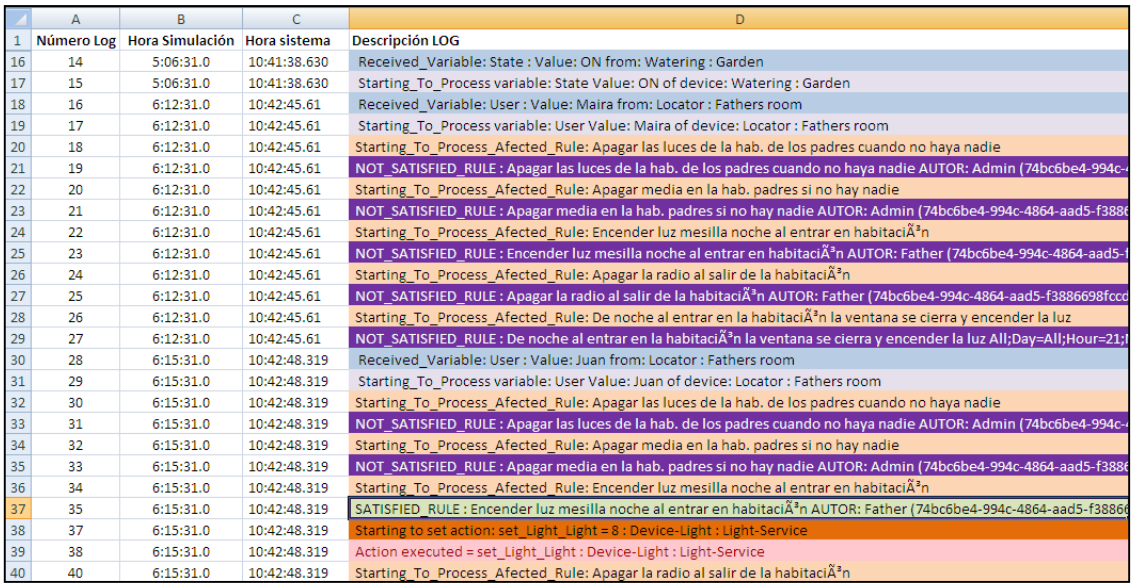
Fig. 7.10 Vista de las opciones para la configuración del reloj

7.6.4 Resultados del Test 2

Todos los resultados de la validación del Test 2 se han guardado en ficheros log. A continuación se mostrarán algunos de esos ficheros tras filtrar la información.

La figura 7.11 muestra el aspecto de un extracto de la lista completa de los *logs* importados a Excel. El número de la primera columna es el número de log asignado secuencialmente por el motor de reglas. En la segunda columna se muestra la hora o momento del día en el que se ejecutó el evento de regla satisfecha (Hora Simulación). En la tercera columna se almacena el momento real en el que se ha ejecutado el evento (Hora sistema). Este tiempo real tiene sentido en simulaciones que se han realizado en 24 minutos para verificar la equivalencia de tiempos entre la escala de 24 horas y la de los 24 minutos. Y, por último, en la cuarta o columna D se muestra la descripción del *log*.

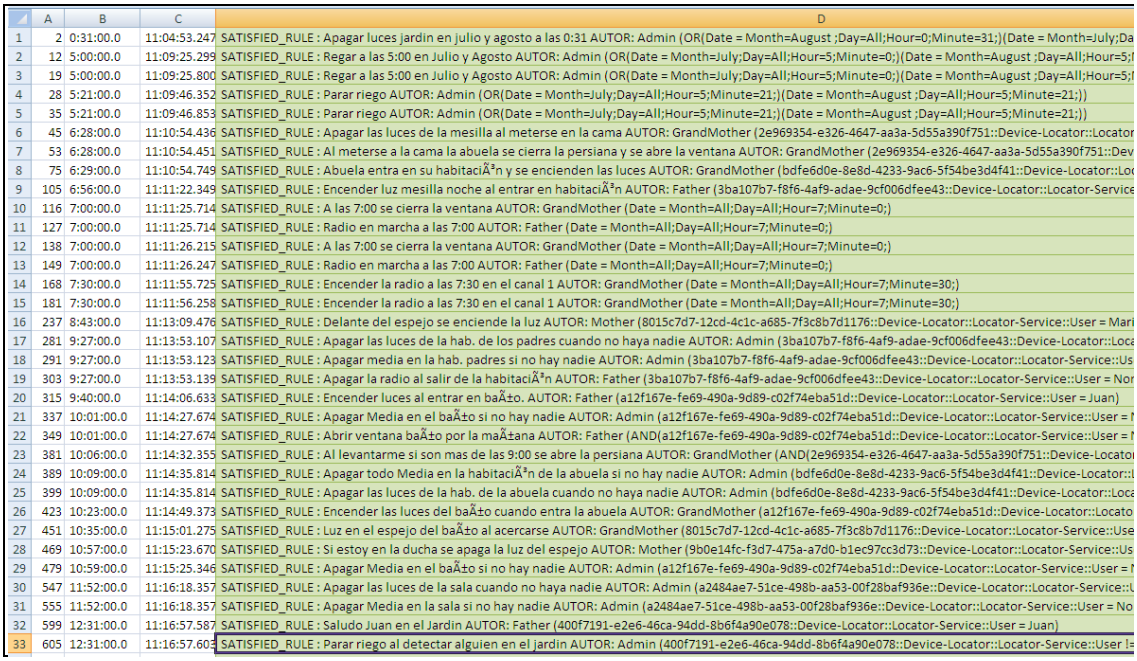
Todas las figuras que se muestran en esta sección siguen esta distribución de columnas.



	A	B	C	D
1	Número Log	Hora Simulación	Hora sistema	Descripción LOG
16	14	5:06:31.0	10:41:38.630	Received_Variable: State : Value: ON from: Watering : Garden
17	15	5:06:31.0	10:41:38.630	Starting_To_Process variable: State Value: ON of device: Watering : Garden
18	16	6:12:31.0	10:42:45.61	Received_Variable: User : Value: Maira from: Locator : Fathers room
19	17	6:12:31.0	10:42:45.61	Starting_To_Process variable: User Value: Maira of device: Locator : Fathers room
20	18	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Apagar las luces de la hab. de los padres cuando no haya nadie
21	19	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (74bc6be4-994c-
22	20	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Apagar media en la hab. padres si no hay nadie
23	21	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f388
24	22	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Encender luz mesilla noche al entrar en habitaciÃ³n
25	23	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-
26	24	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Apagar la radio al salir de la habitaciÃ³n
27	25	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar la radio al salir de la habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd
28	26	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz
29	27	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz All;Day=All;Hour=21;
30	28	6:15:31.0	10:42:48.319	Received_Variable: User : Value: Juan from: Locator : Fathers room
31	29	6:15:31.0	10:42:48.319	Starting_To_Process variable: User Value: Juan of device: Locator : Fathers room
32	30	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Apagar las luces de la hab. de los padres cuando no haya nadie
33	31	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (74bc6be4-994c-
34	32	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Apagar media en la hab. padres si no hay nadie
35	33	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f388
36	34	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Encender luz mesilla noche al entrar en habitaciÃ³n
37	35	6:15:31.0	10:42:48.319	SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f38866
38	37	6:15:31.0	10:42:48.319	Starting to set action: set_Light_Light = 8 : Device-Light : Light-Service
39	38	6:15:31.0	10:42:48.319	Action executed = set_Light_Light : Device-Light : Light-Service
40	40	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Apagar la radio al salir de la habitaciÃ³n

Fig. 7.11 Vista de los *logs* importados en Excel

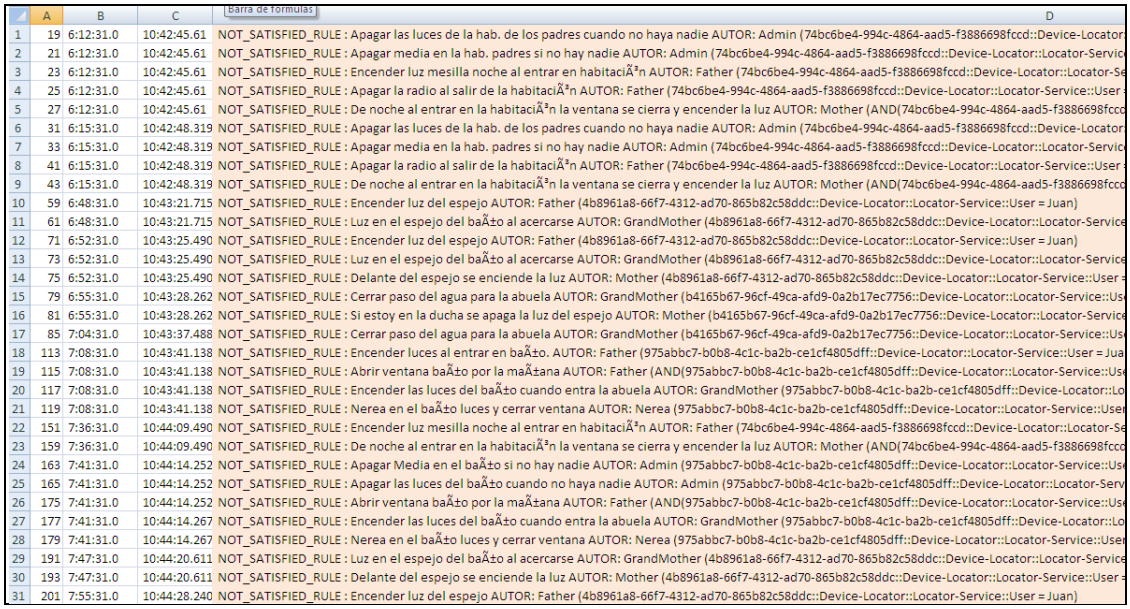
En la figura 7.12 se muestra la lista de las reglas satisfechas en el periodo de un día para el grupo de reglas definidas (apartado 7.6.2).



	A	B	C	D
1	2	0:31:00.0	11:04:53.247	SATISFIED_RULE : Apagar luces jardin en julio y agosto a las 0:31 AUTOR: Admin (OR(Date = Month=August ;Day=All;Hour=0;Minute=31;)(Date = Month=July;Da
2	12	5:00:00.0	11:09:25.299	SATISFIED_RULE : Regar a las 5:00 en Julio y Agosto AUTOR: Admin (OR(Date = Month=July;Day=All;Hour=5;Minute=0;)(Date = Month=August ;Day=All;Hour=5;
3	19	5:00:00.0	11:09:25.800	SATISFIED_RULE : Regar a las 5:00 en Julio y Agosto AUTOR: Admin (OR(Date = Month=July;Day=All;Hour=5;Minute=0;)(Date = Month=August ;Day=All;Hour=5;
4	28	5:21:00.0	11:09:46.352	SATISFIED_RULE : Parar riego AUTOR: Admin (OR(Date = Month=July;Day=All;Hour=5;Minute=21;)(Date = Month=August ;Day=All;Hour=5;Minute=21;))
5	35	5:21:00.0	11:09:46.853	SATISFIED_RULE : Parar riego AUTOR: Admin (OR(Date = Month=July;Day=All;Hour=5;Minute=21;)(Date = Month=August ;Day=All;Hour=5;Minute=21;))
6	45	6:28:00.0	11:10:54.436	SATISFIED_RULE : Apagar las luces de la mesilla al meterse en la cama AUTOR: GrandMother (2e969354-e326-4647-aa3a-5d55a390f751::Device-Locator::Locator
7	53	6:28:00.0	11:10:54.451	SATISFIED_RULE : Al meterse a la cama la abuela se cierra la persiana y se abre la ventana AUTOR: GrandMother (2e969354-e326-4647-aa3a-5d55a390f751::Dev
8	75	6:29:00.0	11:10:54.749	SATISFIED_RULE : Abuela entra en su habitaciÃ³n y se encienden las luces AUTOR: GrandMother (bdfe6d0e-8e8d-4233-9ac6-5f54be3d4f41::Device-Locator::Lo
9	105	6:56:00.0	11:11:22.349	SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (3ba107b7-f8f6-4af9-adae-9cf006dfee43::Device-Locator::Locator-Service
10	116	7:00:00.0	11:11:25.714	SATISFIED_RULE : A las 7:00 se cierra la ventana AUTOR: GrandMother (Date = Month=All;Day=All;Hour=7;Minute=0;)
11	127	7:00:00.0	11:11:25.714	SATISFIED_RULE : Radio en marcha a las 7:00 AUTOR: Father (Date = Month=All;Day=All;Hour=7;Minute=0;)
12	138	7:00:00.0	11:11:26.215	SATISFIED_RULE : A las 7:00 se cierra la ventana AUTOR: GrandMother (Date = Month=All;Day=All;Hour=7;Minute=0;)
13	149	7:00:00.0	11:11:26.247	SATISFIED_RULE : Radio en marcha a las 7:00 AUTOR: Father (Date = Month=All;Day=All;Hour=7;Minute=0;)
14	168	7:30:00.0	11:11:55.725	SATISFIED_RULE : Encender la radio a las 7:30 en el canal 1 AUTOR: GrandMother (Date = Month=All;Day=All;Hour=7;Minute=30;)
15	181	7:30:00.0	11:11:56.258	SATISFIED_RULE : Encender la radio a las 7:30 en el canal 1 AUTOR: GrandMother (Date = Month=All;Day=All;Hour=7;Minute=30;)
16	237	8:43:00.0	11:13:09.476	SATISFIED_RULE : Delante del espejo se enciende la luz AUTOR: Mother (8015c7d7-12cd-4c1c-a685-7f3c8b7d1176::Device-Locator::Locator-Service::User = Mari
17	281	9:27:00.0	11:13:53.107	SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (3ba107b7-f8f6-4af9-adae-9cf006dfee43::Device-Locator::Loca
18	291	9:27:00.0	11:13:53.123	SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (3ba107b7-f8f6-4af9-adae-9cf006dfee43::Device-Locator::Locator-Service::Us
19	303	9:27:00.0	11:13:53.139	SATISFIED_RULE : Apagar la radio al salir de la habitaciÃ³n AUTOR: Father (3ba107b7-f8f6-4af9-adae-9cf006dfee43::Device-Locator::Locator-Service::User = Nor
20	315	9:40:00.0	11:14:06.633	SATISFIED_RULE : Encender luces al entrar en baÃ±o. AUTOR: Father (a12f167e-fe69-490a-9d89-c02f74eba51d::Device-Locator::Locator-Service::User = Juan)
21	337	10:01:00.0	11:14:27.674	SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (a12f167e-fe69-490a-9d89-c02f74eba51d::Device-Locator::Locator-Service::User =
22	349	10:01:00.0	11:14:27.674	SATISFIED_RULE : Abrir ventana baÃ±o por la maÃ±ana AUTOR: Father (AND(a12f167e-fe69-490a-9d89-c02f74eba51d::Device-Locator::Locator-Service::User =
23	381	10:06:00.0	11:14:32.355	SATISFIED_RULE : Al levantarme si son mas de las 9:00 se abre la persiana AUTOR: GrandMother (AND(2e969354-e326-4647-aa3a-5d55a390f751::Device-Locato
24	389	10:09:00.0	11:14:35.814	SATISFIED_RULE : Apagar todo Media en la habitaciÃ³n de la abuela si no hay nadie AUTOR: Admin (bdfe6d0e-8e8d-4233-9ac6-5f54be3d4f41::Device-Locator::L
25	399	10:09:00.0	11:14:35.814	SATISFIED_RULE : Apagar las luces de la hab. de la abuela cuando no haya nadie AUTOR: Admin (bdfe6d0e-8e8d-4233-9ac6-5f54be3d4f41::Device-Locator::Loca
26	423	10:23:00.0	11:14:49.373	SATISFIED_RULE : Encender las luces del baÃ±o cuando entra la abuela AUTOR: GrandMother (a12f167e-fe69-490a-9d89-c02f74eba51d::Device-Locator::Locato
27	451	10:35:00.0	11:15:01.275	SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (8015c7d7-12cd-4c1c-a685-7f3c8b7d1176::Device-Locator::Locator-Service::Use
28	469	10:57:00.0	11:15:23.670	SATISFIED_RULE : Si estoy en la ducha se apaga la luz del espejo AUTOR: Mother (9b0e14fc-f3d7-475a-a7d0-b1ec97cc3d73::Device-Locator::Locator-Service::Us
29	479	10:59:00.0	11:15:25.346	SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (a12f167e-fe69-490a-9d89-c02f74eba51d::Device-Locator::Locator-Service::User =
30	547	11:52:00.0	11:16:18.357	SATISFIED_RULE : Apagar las luces de la sala cuando no haya nadie AUTOR: Admin (a2484ae7-51ce-498b-aa53-00f28baf936e::Device-Locator::Locator-Service::L
31	555	11:52:00.0	11:16:18.357	SATISFIED_RULE : Apagar Media en la sala si no hay nadie AUTOR: Admin (a2484ae7-51ce-498b-aa53-00f28baf936e::Device-Locator::Locator-Service::User = No
32	599	12:31:00.0	11:16:57.587	SATISFIED_RULE : Saludo Juan en el Jardin AUTOR: Father (400f7191-e2e6-46ca-94dd-8b6f4a90e078::Device-Locator::Locator-Service::User = Juan)
33	605	12:31:00.0	11:16:57.603	SATISFIED_RULE : Parar riego al detectar alguien en el Jardin AUTOR: Admin (400f7191-e2e6-46ca-94dd-8b6f4a90e078::Device-Locator::Locator-Service::User !=

Fig. 7.12 Lista de los *logs* donde se han evaluado positivamente las reglas de los usuarios

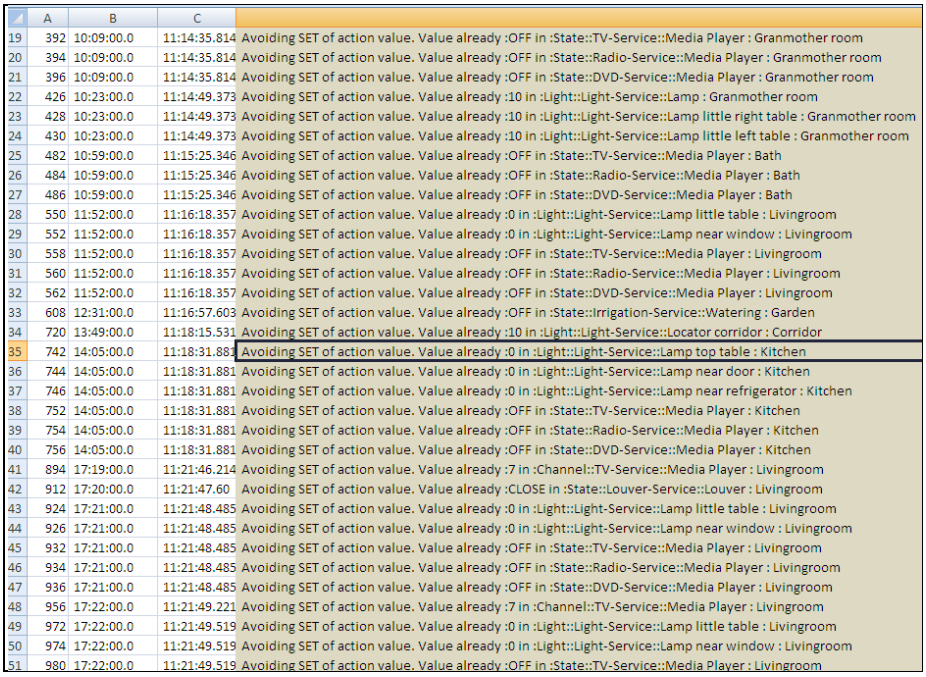
En la figura 7.13 se muestra la lista de las reglas no satisfechas en el periodo de un día para el grupo de reglas definidos.



	A	B	C	D
1	19	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator
2	21	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Servic
3	23	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Se
4	25	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar la radio al salir de la habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Service::User =
5	27	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz AUTOR: Mother (AND(74bc6be4-994c-4864-aad5-f3886698fccd
6	31	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator
7	33	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Servic
8	41	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar la radio al salir de la habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Service::User =
9	43	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz AUTOR: Mother (AND(74bc6be4-994c-4864-aad5-f3886698fccd
10	59	6:48:31.0	10:43:21.715	NOT_SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Juan)
11	61	6:48:31.0	10:43:21.715	NOT_SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service
12	71	6:52:31.0	10:43:25.490	NOT_SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Juan)
13	73	6:52:31.0	10:43:25.490	NOT_SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service
14	75	6:52:31.0	10:43:25.490	NOT_SATISFIED_RULE : Delante del espejo se enciende la luz AUTOR: Mother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User =
15	79	6:55:31.0	10:43:28.262	NOT_SATISFIED_RULE : Cerrar paso del agua para la abuela AUTOR: GrandMother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::Us
16	81	6:55:31.0	10:43:28.262	NOT_SATISFIED_RULE : Si estoy en la ducha se apaga la luz del espejo AUTOR: Mother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service
17	85	7:04:31.0	10:43:37.488	NOT_SATISFIED_RULE : Cerrar paso del agua para la abuela AUTOR: GrandMother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::Us
18	113	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Encender luces al entrar en baÃ±o. AUTOR: Father (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = Jua
19	115	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Abrir ventana baÃ±o por la maÃ±ana AUTOR: Father (AND(975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::Us
20	117	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Encender las luces del baÃ±o cuando entra la abuela AUTOR: GrandMother (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Lo
21	119	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Nerea en el baÃ±o luces y cerrar ventana AUTOR: Nerea (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User
22	151	7:36:31.0	10:44:09.490	NOT_SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Se
23	159	7:36:31.0	10:44:09.490	NOT_SATISFIED_RULE : De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz AUTOR: Mother (AND(74bc6be4-994c-4864-aad5-f3886698fccd
24	163	7:41:31.0	10:44:14.252	NOT_SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::Us
25	165	7:41:31.0	10:44:14.252	NOT_SATISFIED_RULE : Apagar las luces del baÃ±o cuando no haya nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Serv
26	175	7:41:31.0	10:44:14.252	NOT_SATISFIED_RULE : Abrir ventana baÃ±o por la maÃ±ana AUTOR: Father (AND(975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::Us
27	177	7:41:31.0	10:44:14.267	NOT_SATISFIED_RULE : Encender las luces del baÃ±o cuando entra la abuela AUTOR: GrandMother (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Lo
28	179	7:41:31.0	10:44:14.267	NOT_SATISFIED_RULE : Nerea en el baÃ±o luces y cerrar ventana AUTOR: Nerea (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User
29	191	7:47:31.0	10:44:20.611	NOT_SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service
30	193	7:47:31.0	10:44:20.611	NOT_SATISFIED_RULE : Delante del espejo se enciende la luz AUTOR: Mother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User =
31	201	7:55:31.0	10:44:28.240	NOT_SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Juan)

Fig. 7.13 Lista de *logs* en el que las reglas han sido evaluadas pero no satisfechas

En la figura 7.14 por ejemplo se muestra la lista de acciones no ejecutadas, por tener de antemano el valor deseado en el periodo de un día para el grupo de reglas definidos.



	A	B	C	
19	392	10:09:00.0	11:14:35.814	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Media Player : Granmother room
20	394	10:09:00.0	11:14:35.814	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Media Player : Granmother room
21	396	10:09:00.0	11:14:35.814	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Media Player : Granmother room
22	426	10:23:00.0	11:14:49.373	Avoiding SET of action value. Value already :10 in :Light::Light-Service::Lamp : Granmother room
23	428	10:23:00.0	11:14:49.373	Avoiding SET of action value. Value already :10 in :Light::Light-Service::Lamp little right table : Granmother room
24	430	10:23:00.0	11:14:49.373	Avoiding SET of action value. Value already :10 in :Light::Light-Service::Lamp little left table : Granmother room
25	482	10:59:00.0	11:15:25.346	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Media Player : Bath
26	484	10:59:00.0	11:15:25.346	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Media Player : Bath
27	486	10:59:00.0	11:15:25.346	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Media Player : Bath
28	550	11:52:00.0	11:16:18.357	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp little table : Livingroom
29	552	11:52:00.0	11:16:18.357	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp near window : Livingroom
30	558	11:52:00.0	11:16:18.357	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Media Player : Livingroom
31	560	11:52:00.0	11:16:18.357	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Media Player : Livingroom
32	562	11:52:00.0	11:16:18.357	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Media Player : Livingroom
33	608	12:31:00.0	11:16:57.603	Avoiding SET of action value. Value already :OFF in :State::Irrigation-Service::Watering : Garden
34	720	13:49:00.0	11:18:15.531	Avoiding SET of action value. Value already :10 in :Light::Light-Service::Locator corridor : Corridor
35	742	14:05:00.0	11:18:31.881	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp top table : Kitchen
36	744	14:05:00.0	11:18:31.881	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp near door : Kitchen
37	746	14:05:00.0	11:18:31.881	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp near refrigerator : Kitchen
38	752	14:05:00.0	11:18:31.881	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Media Player : Kitchen
39	754	14:05:00.0	11:18:31.881	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Media Player : Kitchen
40	756	14:05:00.0	11:18:31.881	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Media Player : Kitchen
41	894	17:19:00.0	11:21:46.214	Avoiding SET of action value. Value already :7 in :Channel::TV-Service::Media Player : Livingroom
42	912	17:20:00.0	11:21:47.60	Avoiding SET of action value. Value already :CLOSE in :State::Louver-Service::Louver : Livingroom
43	924	17:21:00.0	11:21:48.485	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp little table : Livingroom
44	926	17:21:00.0	11:21:48.485	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp near window : Livingroom
45	932	17:21:00.0	11:21:48.485	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Media Player : Livingroom
46	934	17:21:00.0	11:21:48.485	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Media Player : Livingroom
47	936	17:21:00.0	11:21:48.485	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Media Player : Livingroom
48	956	17:22:00.0	11:21:49.221	Avoiding SET of action value. Value already :7 in :Channel::TV-Service::Media Player : Livingroom
49	972	17:22:00.0	11:21:49.519	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp little table : Livingroom
50	974	17:22:00.0	11:21:49.519	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Lamp near window : Livingroom
51	980	17:22:00.0	11:21:49.519	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Media Player : Livingroom

Fig. 7.14 Lista de los *logs* donde las acciones no se ejecutaron, dado que el efecto de las acciones se daba en el entorno

Por último, en la figura 7.15 se muestran como ejemplo los conflictos detectados y como se han resuelto. Por ejemplo, en la línea 6 de la figura 7.15, se puede ver como una regla se antepone a la otra. Al cumplirse las condiciones de la regla de sistema, el valor de la acción original es modificado.

	A	B	C	D	E
1	349	9:09:31.0	10:45:42.659	Starting to evaluate Afected_OTHER_USER Rule: Encender luz del espejo ->set_Light_Light = 10	0,0004126170
2	352	9:09:31.0	10:45:42.659	NOT_Afected_OTHER_USER_Rule	0,0005032895
3	635	11:15:31.0	10:47:49.334	Starting to evaluate Afected_OTHER_USER Rule: Subir TÂª casa si baja la media de 20Â ºC ->set_Heater_Temperature = 22	0,0020715813
4	638	11:15:31.0	10:47:49.334	NOT_Afected_OTHER_USER_Rule	0,0021334843
5	639	11:15:31.0	10:47:49.334	Starting to evaluate AFECTED_MANDATORY_RULE: set_Heater_Temperature = 22	0,0021506750
6	642	11:15:31.0	10:47:49.334	Afected: Applying_Mandatory_Rule over User's one.set_Heater_Temperature = 24	0,0024261570
7	645	11:15:31.0	10:47:49.334	Starting to evaluate Afected_OTHER_USER Rule: TV canal 2 entre las 15:00 y las 16:00 al sentarme en el sofa ->set_TV_State = ON	0,0042117001
8	648	11:15:31.0	10:47:49.334	NOT_Afected_OTHER_USER_Rule	0,0044134809
9	651	11:15:31.0	10:47:49.334	Starting to evaluate Afected_OTHER_USER Rule: TV canal 2 entre las 15:00 y las 16:00 al sentarme en el sofa ->set_TV_Channel = 8	0,0044858889
10	654	11:15:31.0	10:47:49.334	NOT_Afected_OTHER_USER_Rule	0,0046654745
11	697	11:44:31.0	10:48:18.359	Starting to evaluate Afected_OTHER_USER Rule: Apagar Media en la cocina si no hay nadie ->set_TV_State = ON	0,0051923095
12	700	11:44:31.0	10:48:18.359	NOT_Afected_OTHER_USER_Rule	0,0055222179
13	703	11:44:31.0	10:48:18.359	Starting to evaluate Afected_OTHER_USER Rule: Por la maÃ±ana en la cocina al entrar TV 5 ->set_TV_Channel = 10	0,0075777423
14	706	11:44:31.0	10:48:18.375	NOT_Afected_OTHER_USER_Rule	0,0079791144
15	809	12:19:31.0	10:49:20.921	Starting to evaluate Afected_OTHER_USER Rule: Nerea luces por el pasillo por la noche ->set_Light_Light = 10	0,0048294843
16	812	12:19:31.0	10:49:20.968	NOT_Afected_OTHER_USER_Rule	0,0475083454

Fig. 7.15 Vista de las reglas en conflicto

La figura 7.16 muestra un resumen de los resultados del Test 2. En total, se ha generado de la ejecución de 61 reglas en el motor de reglas y 79 acciones en el 'action feeder' (figura 7.16). Los resultados no pretenden mostrar ningún tipo de tendencia ni comportamiento. Con otras reglas, los resultados serían totalmente diferentes. No obstante, han servido para verificar que, dado un conjunto importante y significativo de eventos-reglas-acciones, el comportamiento del motor de reglas es correcto.

Número de eventos recibidos	153
Número de reglas evaluadas	222
Reglas evaluadas y no ejecutadas	175
Reglas evaluadas y ejecutadas	47
Acciones para ejecutar	93
Acciones ejecutadas	32
Ejecución de acciones descartadas (mismo valor)	61
Reglas en conficto	8
Ejecución de reglas después de entrar en conflicto por prioridad	1

Fig. 7.16 Resumen de los resultados del Test 2

De las 79 acciones simuladas (que provocan su correspondiente evento), se han recibido un total de 153 eventos. La diferencia se explica porque la ejecución de las acciones de las reglas a su vez desencadenará más eventos. Se puede ver que 32 acciones han sido ejecutadas, que provocan 32 eventos, y con lo anterior suman 111. Sin embargo, el número de eventos recibidos fue mayor de 111. La explicación está en que los dispositivos de luz, para pasar del estado encendido al apagado, pasan por estados intermedios y el MR los analiza como nuevos eventos. Estos dispositivos simulan un variador de intensidad de luz, por lo que para pasar de 0 (apagado) a 10 (encendido) pasan por los valores intermedios de 2,4,6 y 8, siendo estos cambios también recibidos como eventos en el motor de reglas. Este tipo de eventos es el que provoca también la evaluación de reglas que no son ejecutadas. Finalmente, comentar que se han realizado 32 ejecuciones de acciones, frente a 61 acciones que no han sido ejecutadas por tener ya el valor anteriormente. La ejecución de una acción se suspende, ya que el MR verifica antes de cualquier modificación si el valor es diferente al existente. En caso de que los valores sean iguales, se aborta el proceso de la ejecución de la acción evitando la llamada y el envío de los mensajes pertinentes a la red.

En el 'Anexo E Resultados de test', se muestran los resultados anteriores con más detalle.

7.7 Test 3: Estudio del funcionamiento del motor de reglas cuando los dispositivos desaparecen inesperadamente en la red

El MR incorpora una utilidad que comprueba dinámicamente la existencia de los dispositivos en la red con los que interactúa. El MR está dotado de un mecanismo de 'ping' personalizado. Se utiliza un ping para comprobar si un dispositivo host está disponible, enviando paquetes o mensajes de control para comprobar si el dispositivo host está activo. El mecanismo de ping se hace necesario para garantizar el buen funcionamiento de la plataforma en un entorno cambiante en el que los dispositivos aparecen y desaparecen inesperadamente en la red. Esto es muy importante para el caso de las comunicaciones inalámbricas (no tan robustas como las cableadas) donde se producen desconexiones temporales que deberían ser detectadas. También puede ocurrir que los dispositivos empotrados con recursos limitados se queden inactivos para ahorrar energía. Cuando el MR detecta un fallo de comunicaciones o desconexión con algún dispositivo, las variables de dichos dispositivos que forman parte de una o varias reglas definidas pasan a modo 'pasivo'. Que el valor de una variable pase a ser pasivo significa que dicho valor se corresponde con uno por defecto, indicado por el usuario en el momento de la creación de la condición o con el último valor conocido. Así pues, si al producirse un evento se tuviera que evaluar la variable de un dispositivo 'no activo', este valor nunca sería nulo evitando provocar un error.

7.7.1 Descripción del test

El test mostrará el comportamiento del MR cuando se produce la desconexión de dos dispositivos, como son, una PDA y un dispositivo virtual ejecutado en uno de los PCs. El dispositivo móvil (PDA) lo alejaremos de tal manera de su punto de acceso provocando su desconexión, y en cuanto al PC, simplemente lo desconectaremos de la red. Toda la información sobre las conexiones y desconexiones ocurridas se guardan en un fichero para su posterior análisis.

Primero, se tiene que activar el mecanismo de ping en el editor del MR. La figura 7.17 muestra la activación de dicho mecanismo. Basta con simplemente verificar el campo de texto 'Check if alive' en la interfaz del MR.

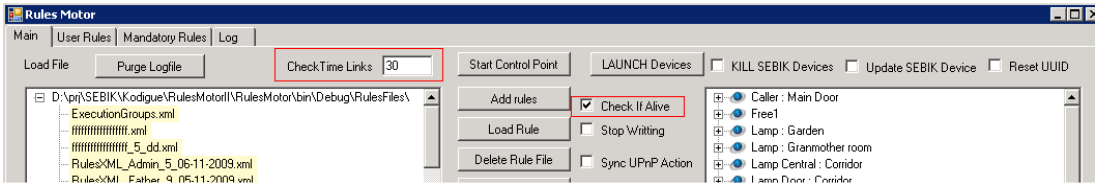


Fig. 7.17 Vista parámetros configuración ping

A continuación, se puede configurar el tiempo de ejecución del *ping* en el campo 'Check Time Links'. En este caso, el tiempo de frecuencia es de 30 segundos. De esta forma, cada 30 segundos se verificará la conexión con cada dispositivo. Normalmente, por cada dispositivo se realiza una llamada de *ping* síncrona, dado que en condiciones normales el MR tarda del orden de milisegundos en recibir el mensaje de respuesta.

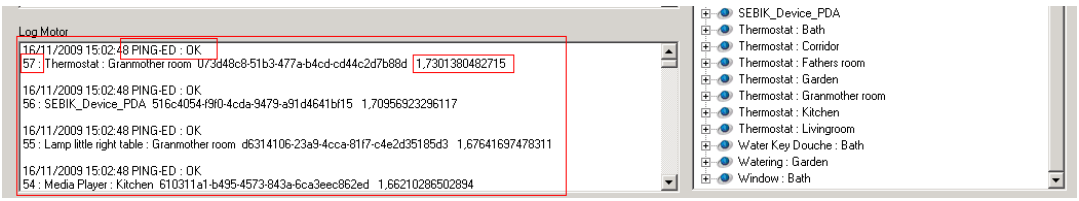


Fig. 7.18 Vista *log* de *ping* correcto enviado por el MR

Como se puede ver en la figura 7.18, se ha producido un *ping* correcto a los 57 dispositivos que tiene en su lista el motor de reglas. En este caso se han necesitado 1,73 segundos para realizar todos los pings a todos los dispositivos. Este tiempo de respuesta varía cada vez, ya que depende de factores externos como el ancho de banda de la red, su rendimiento y de los propios dispositivos.

A continuación, vamos a provocar el fallo de comunicación, alejando la PDA del punto de acceso y desconectando el PC de la red, tal y como se ha explicado anteriormente.

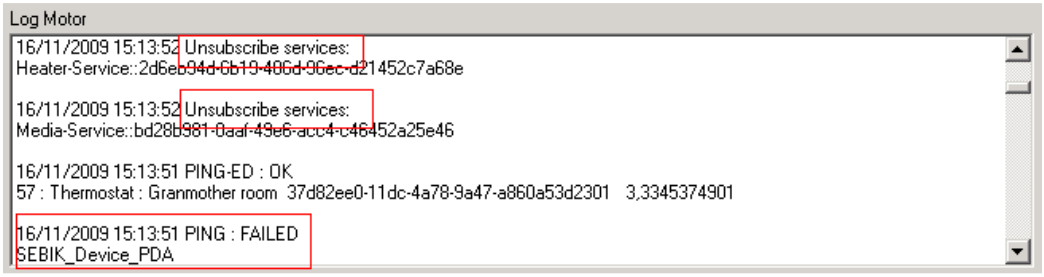


Fig. 7.19 Vista *log* de ping fallido

En la Fig. 7.19, se muestra que después de una llamada de *ping* fallida comienza el proceso de de-suscripción de los servicios del dispositivo PDA. Mientras el proceso de chequeo de ping continua con el resto de los dispositivos, en paralelo comienza la de-suscripción de todos los servicios del dispositivo no localizado. Dependiendo de lo que el usuario ha decidido a la hora de crear las reglas, los valores de las variables pasarán a tener el valor por defecto, tal y como se muestra en la figura 7.20.

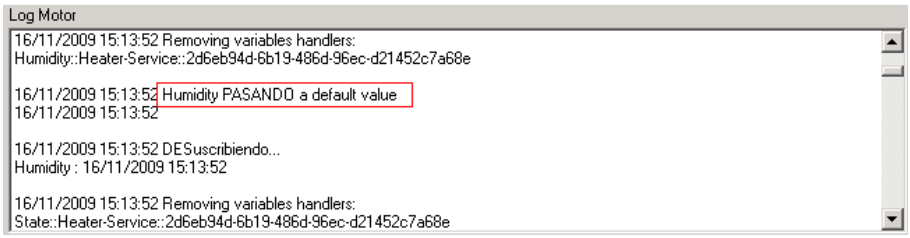


Fig. 7.20 Vista log de-suscripción de variable

La próxima vez que el dispositivo vuelva a conectarse a la red, éste se anunciará utilizando los canales normales descritos en el apartado 5.4.1.1, y a posteriori el MR lo incluirá en su lista de dispositivos como si fuera la primera vez, restaurando todos los enlaces con los servicios y variables existentes.

7.8 Test 4: Usabilidad del editor de reglas

Para finalizar con los test de validación, se va a realizar un test de la usabilidad del editor de reglas con la ayuda de varios participantes. Se trata de que dichos participantes (que desconocen el sistema), después de recibir una explicación breve de su funcionamiento, traten de crear tres o cuatro reglas de comportamiento de un hogar en el editor, y que a continuación respondan a un pequeño cuestionario para obtener una evaluación respecto a la usabilidad del editor de reglas.

7.8.1 Descripción

El número de test realizados fue de diez, con cuatro mujeres y seis hombres de edades entre 23 y 43 años. Entre los participantes existentes, se encuentran desarrolladores de aplicaciones, técnicos gestores de proyectos, administrativos, secretarias y comerciales, todos ellos al menos con nociones básicas en la utilización de ordenadores personales y/o dispositivos electrónicos. En el 'Anexo D' se incluye el cuestionario utilizado y el proceso seguido para la realización de la misma. Las reglas de ejemplo que tuvieron que crear los usuarios fueron:

Regla 1: Cuando el usuario 'Juan' entre en la cocina, que se enciendan las luces y se abra la persiana.

Regla 2: Si la temperatura media de la sala, cocina y pasillo es menor que 18ºC, poner la calefacción en marcha a una temperatura de 22ºC en la sala y cocina, y a 20ºC en el pasillo.

Regla 3: Si el usuario 'Nerea' entra en la sala entre las 17:00 y 20:00, que se ponga la televisión en marcha en el canal 5, con el volumen al 40%. Que se atenúen todas las luces de la sala también.

Regla 4: Cuando no haya 'nadie' en casa que se apaguen todas las luces.

7.8.2 Resultados y conclusiones

A continuación se describen los resultados más importantes extraídos de la realización de las pruebas.

7.8.2.1 Resultados

Tal y como era de esperar, las reglas 1 y 2 fueron creadas sin problemas por todos los usuarios. Sin embargo, no ocurrió lo mismo en la creación de las reglas 3 y 4. Dichas reglas implicaban la creación de reglas de tipo grupo (recordar que este tipo de reglas sirve para agrupar todos los dispositivos que ofrecen un mismo servicio. Mas detalle

en el apartado 6.1.2.3) y debido a esta peculiaridad, algunas personas (7 personas) necesitaron ayuda para la creación de las mismas. De todas formas, todas las personas han sido capaces de identificar que algo 'diferente' (condiciones de tipo grupo) había en las reglas 3 y 4.

De lo anterior se deduce que se necesita una fase de aprendizaje en la creación de reglas complejas. De todas formas, parece que una vez que se muestra la dinámica del editor de reglas, los usuarios son capaces de crear reglas y se sienten cómodos con el editor según se muestra en los resultados de los cuestionarios.

El tiempo transcurrido en la creación de las reglas ha variado en todos los casos, tardando entre 12 y 25 minutos en completar la prueba por los diferentes usuarios. Se cree que una vez familiarizado con el editor de reglas, cualquier usuario podría crear las reglas para definir el comportamiento en un entorno en un tiempo razonable.

7.8.2.2 Comentarios de los usuarios

En esta sección se describen los principales comentarios de los usuarios respecto a la personalización del hogar y el editor de reglas. Estas aportaciones son de gran utilidad para valorar la calidad del editor con respecto a su usabilidad y practicabilidad.

I. Respecto a la personalización del hogar mediante reglas, se señalaron los siguientes comentarios:
    a. Mejora del confort y de la eficiencia energética.
    b. Ofrece una personalización mayor que una simple programación horaria de los electrodomésticos en el hogar.
    c. Gran interés en el apagado de luces automático cuando alguien no está en una habitación.
    d. Se resaltó la utilidad de todas las reglas en general, algunas por practicidad y otras por confort.

II. Respecto al entorno gráfico o al interfaz del editor se hicieron los siguientes comentarios:
    a. Reducción del número de reglas: demasiados tipos de reglas (temporizadas, de agrupación, etc.).
    b. Gran interés en la posibilidad de creación de condiciones o acciones de grupo.
    c. Sin una breve explicación cuesta darse cuenta del concepto de dispositivo o servicio.
    d. Adición de un asistente que te guie y explique cómo crear las reglas.
    e. Debería tener iconos o botones para 'agregar condición', 'eliminar,…' en vez de tener que utilizar el ratón para desplegar del menú.
    f. Selección más fácil de los dispositivos, pudiendo filtrarlos por zonas o tipo.
    g. Gestión y supervisión desde internet.

h. A la hora de crear una acción que aparezcan resaltados los servicios utilizables. Que los elementos sean claramente visibles y atractivos (una imagen vale más que mil palabras): ON (en verde), OFF (en rojo), temperatura (todos en azul)…

i. Ordenar los elementos alfabéticamente para realizar búsquedas. Utilizaría botones con imágenes, colores…

j. Cuando hay que colocarse encima y pulsar con el botón derecho, que aparezca un comentario (ayuda opcional), si no se olvida.

7.8.2.3 Conclusiones de la validación con usuarios

De los comentarios anteriores, respecto al entorno gráfico utilizado se puede concluir que a los usuarios en general la forma de seleccionar dispositivos en una lista desplegable les ha resultado incómoda. En este sentido, la utilización de botones más grandes, y otros mecanismos de ordenación y clasificación parece que facilitaría la creación de reglas. La utilización de asistentes y ayuda contextual es otra característica interesante que parece ser demandada por los usuarios.

En cualquier caso, el objetivo principal de la validación con usuarios era estudiar la usabilidad de la personalización del hogar y asimismo, valorar la funcionalidad del editor de reglas. Como en todos los diseños iniciales de una interfaz gráfica, las mejoras vienen de las sugerencias de los usuarios. Tras esta validación, se ven claramente las mejoras que podrían ser incorporadas en dicho editor para facilitar la creación de las reglas con un editor de reglas más acorde con los gustos de todo el mundo.

Por otra parte, señalar que todos los usuarios calificaron que la personalización del hogar sería algo muy útil.

7.9 Conclusiones

En este capítulo, se muestran los resultados de la validación, tanto del editor de reglas como del motor de reglas. Los resultados relativos al motor de reglas muestran su correcto funcionamiento y su buen rendimiento ante condiciones adversas. Primero se analizó su rendimiento al desencadenarse un número elevado de acciones. En la figura 7.21 se muestra la vista con los valores de todos los test realizados. La media del tiempo de respuesta en todos los casos, incluso para un número elevado de reglas, está por debajo del segundo.

En segundo lugar, se estudiaron los resultados de su comportamiento, ante un número elevado de eventos y reglas en una simulación de un hogar virtual a lo largo de un día. En la figura 7.22 se resumen los mismos

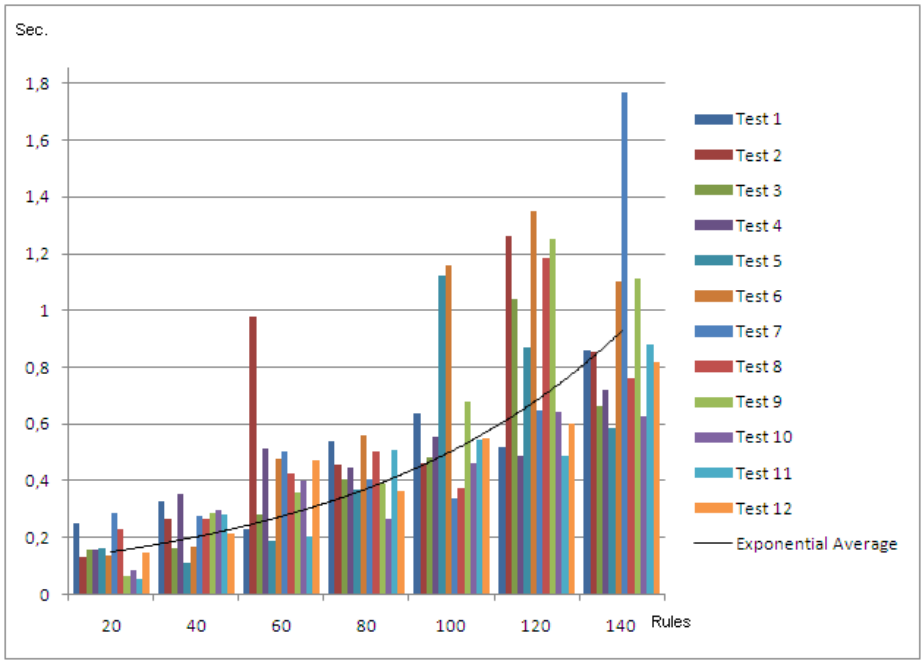


Fig. 7.21 Respuesta en segundos del motor de reglas para cada grupo de reglas

.

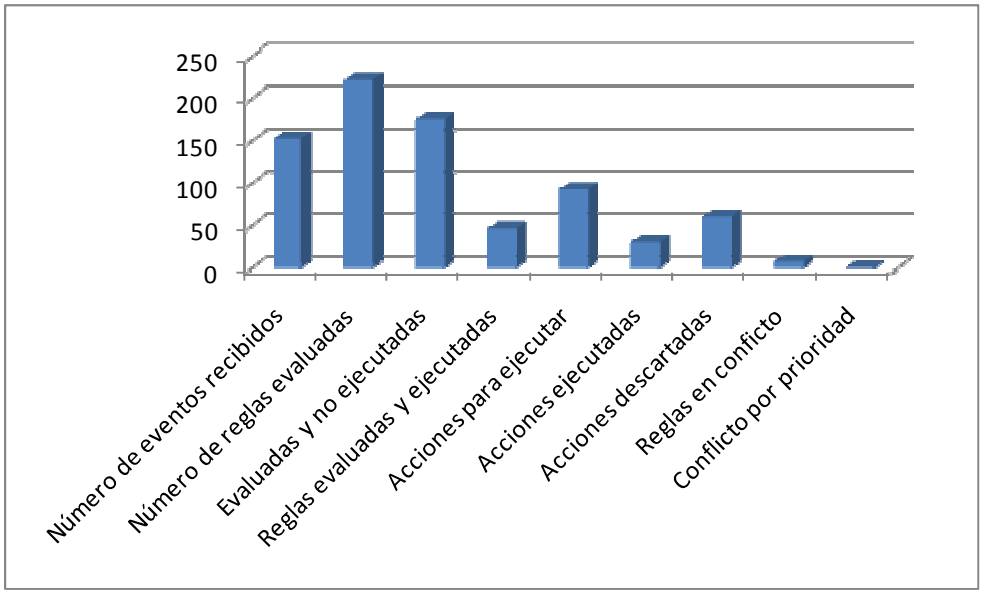


Fig. 7.22 Resumen de los resultados del Test 2

Finalmente, se estudió cómo el *middleware* detectaba la aparición o desaparición de diferentes dispositivos.

Con respecto al editor de reglas para personalizar un entorno, se puede decir que su utilidad fue ampliamente valorada y respaldada por parte de los encuestados, aunque la interfaz debería ser modificada para lograr una interacción más sencilla e intuitiva, y más acorde a lo indicado por los participantes.

Capítulo 8: Aportaciones originales y líneas abiertas

Si bien en cada capítulo se han ido plasmando las conclusiones correspondientes, a continuación, se muestran de tal forma que se pueda observar el grado del cumplimiento de los objetivos marcados y las aportaciones originales resultantes en cada paso.

El trabajo realizado en esta tesis propone una solución integral al problema de la personalización y configuración de entornos ubicuos o inteligentes, y la solución aportada pretende ser utilizada por el usuario final independientemente de su nivel de conocimiento informático.

8.1 Cumplimiento de objetivos y aportaciones

El ***middleware framework CAHIM*** desarrollado contribuye con las siguientes aportaciones:

- **Mecanismos para la interoperabilidad de las aplicaciones y de los dispositivos.** Un primer paso para la configuración de entornos más inteligentes y sensibles al contexto es permitir que los dispositivos desplegados sean descubribles e invocables de una manera uniforme a pesar de su heterogeneidad. Para resolver este común problema de la heterogeneidad (objetivos 1 y 2), el middleware framework propuesto permite el acceso a cualquier dispositivo independientemente del servicio de descubrimiento de servicios. Este acceso garantiza el descubrimiento y la actuación transparente con todo tipo de dispositivos desde las aplicaciones. La solución aportada provee de mecanismos para que las aplicaciones, independientemente del protocolo de descubrimiento de servicios utilizado, puedan hacer uso de los servicios de los dispositivos disponibles en el entorno. Los programadores de estas aplicaciones, que se comunican a través de este middleware, no están obligados a codificar en un cierto lenguaje de programación compatible con las interfaces del middleware, sino que pueden elegir el entorno de programación que mejor se adapte a sus necesidades. Todo esto se lleva a cabo gracias a un modelo unificado de servicios que permite describir los servicios de todos los dispositivos y a unos bridges (o componentes) que actúan como gateways o puertas de enlace para que las aplicaciones puedan instanciar cualquier servicio existente: UPnP services, Jini services, Web Services, etc.

  Además, este nuevo middleware proporciona mecanismos para abordar el dinamismo (objetivo 3) del entorno y la escalabilidad ante la llegada de nuevos dispositivos o protocolos de descubrimiento de servicios. La modificación y/o inclusión de nuevas versiones en los diferentes componentes del middleware está garantizada gracias a las capas ‘Unified Model’ y ‘Bridges’, del

componente de interoperabilidad. Dichas capas garantizan la compatibilidad 'backward & forward' respecto a la modificación y/o inclusión de nuevas versiones en los diferentes componentes del middleware.

- **Mecanismos para la movilidad de aplicaciones.** Nuestro middleware incluye un mecanismo que aporta la capacidad de mover las configuraciones y mantener los estados de las aplicaciones independientemente de la plataforma de ejecución y de los dispositivos que se utilizan, restableciendo las configuraciones de ejecución cada vez que existe un cambio de ubicación (objetivo 4).
- **Mecanismos para dotar al entorno de inteligencia.** La solución adoptada utiliza un motor de reglas ECA para dotar al entorno de cierta inteligencia definida por el usuario (objetivo 5); para la creación de entornos más configurables y personalizables. Este motor soporta el dinamismo característico de los entornos ubicuos y además presenta las siguientes características:
    a. Optimiza el tiempo de evaluación con respecto a los motores de reglas secuenciales
    b. *Optimiza el tiempo de ejecución y* de respuesta**,** dado que se crean espacios de ejecución (threads) independientes para cada condición de la regla. Además, al crearse threads paralelos, el mal funcionamiento o retraso de la ejecución de las acciones de una regla no afectan al sistema. En la validación, se ha observado que el tiempo de respuesta (ejecución de la acción de una regla) es menor a 1 segundo.
    c. Ejecución en dispositivos con recursos limitados como pueden ser una PDA o Smartphone.
    d. Resolución de conflictos en tiempo de ejecución basado en prioridades de los usuarios y las reglas.
- **Herramienta para personalizar el entorno por un usuario final.** Se ha desarrollado un configurador (objetivo 6) en forma de editor gráfico para la generación de reglas ECA por parte del usuario final, denominado TAMAmI. Tras la evaluación de TAMAmI con usuarios, se puede concluir que un usuario sin conocimientos técnicos específicos podría personalizar el comportamiento de su hogar. Además, la herramienta ayuda a que el usuario no sea consciente del middleware subyacente, logrando: (1) que el descubrimiento de los componentes software y hardware sea transparente para el usuario (Plug & Play) y (2) que la movilidad de las aplicaciones entre los distintos elementos de computación para llevar a cabo un escenario 'follow-me' esté garantizada.

  En definitiva, esta herramienta hace accesibles y usables los componentes del *middleware framework* CAHIM resultantes de esta tesis, todos ellos dirigidos hacia una mejor y más inteligente personalización y configuración de los servicios provistos mediante tecnologías heterogéneas en entornos ubicuos.

8.2 Publicaciones relevantes

Las conclusiones y aportaciones que se han ido logrando con este trabajo de investigación, se han ido divulgando en foros científicos de cara a contrastar la validez e interés de los resultados y aportaciones del mismo. Se han realizado presentaciones y publicaciones en foros nacionales (Ucami, CEDI) e internacionales (Intersense, SASO-Perada,…), siendo varios de los artículos indexados. La descripción de las publicaciones relacionadas con el autor y el trabajo descrito a lo largo de esta tesis se encuentran detallados en el Anexo G.

8.3 Líneas abiertas de investigación

A lo largo de este trabajo de investigación, también se han encontrado ciertos aspectos que no siendo objeto de esta tesis, aparecen como retos importantes a tener en cuenta como futuras líneas de investigación. Estas son:

1. **Interoperabilidad.** Hemos visto cómo el *middleware framework* abstrae de los problemas de interoperabilidad en los entornos heterogéneos de servicios, sin embargo existen todavía otros problemas no abordados. Tal y como se ha analizado a lo largo de la tesis, un problema que existe es la no uniformidad o estandarización de las representaciones de los servicios. Así pues, se debería trabajar en una descripción de los servicios de los dispositivos que debería ser uniforme para garantizar un adecuado descubrimiento, composición y gestión de los mismos. Asimismo para que los dispositivos y las aplicaciones sean tanto o más interoperables puede ser la utilización de ontologías descriptoras de servicios. La utilización de descripciones unificadas basadas en ontologías como puede ser OWL-S, podría permitir el modelado de todo tipo de servicios abstractos y concretos siguiendo la estructuración de la información acorde a un modelo semántico.
2. **Composición y coordinación de servicios,** para lograr la creación de servicios más complejos a partir de varios servicios**.** Los servicios utilizados han sido servicios simples; sin embargo, el uso de servicios compuestos, combinando dos o más servicios, mejoraría varios aspectos tales como la generación de nuevas funcionalidades en el entorno. La composición y coordinación de servicios se podría llevar a cabo a través de mecanismos de orquestación o coreografía entre los diferentes componentes del middleware.
3. **Motor de reglas.** La personalización de entornos multiusuario es complicada y todavía queda hacer frente a la compleja temática de resolución de conflictos entre las distintas reglas de los múltiples usuarios. En esta tesis, se ha abordado la resolución de conflictos cuando es un único motor de reglas el que gestiona y evalúa las reglas de varios usuarios. Sin embargo, podría ocurrir que en un mismo entorno coincidieran dos o más motores de reglas, cada uno

de los cuáles evaluase distintas reglas que dieran como resultado situaciones contradictorias.

4. El **aprendizaje** automático por parte del *middleware framework*, y la adaptación continua a los hábitos de uso de los usuarios es otro aspecto susceptible de aplicación y mejora. La modificación dinámica de prioridades de las acciones en base al uso y/o preferencias del usuario permitiría al sistema adaptarse a lo largo del tiempo siendo innecesaria la participación activa del usuario para cambiar el comportamiento inicialmente pre-establecido.
5. La **seguridad y privacidad** son muy importantes en entornos ubicuos, en los que existen una gran variedad de dispositivos que forman redes de forma espontánea sin infraestructura aparente y que se pasan información unos a otros. Surge en consecuencia la necesidad de proveer de algún mecanismo de seguridad a las comunicaciones entre dispositivos como pueden ser mecanismos de autentificación y autorización. Lo primero se podría lograr utilizando cierto tipo de encriptación de datos, y lo segundo utilizando certificados para validar a los dispositivos. No obstante, el dotar de unos mecanismos adecuados de privacidad y seguridad no es trivial, ya que se contrapone directamente a la filosofía del 'Plug&Play' o de las redes espontáneas de dispositivos.
6. La **utilización de la red internet** como plataforma operativa y proveedora de servicios, es un aspecto innovador para este tipo de sistemas. La tarificación por el consumo de servicios ubicuos que hayan sido ejecutados en entornos físicos fuera de nuestro hogar u oficina es otro factor de interés tanto para los proveedores de servicios como para los operadores de telecomunicaciones. La provisión de nuevos servicios ajenos al middleware (por ejemplo, noticias, estado de las carreteras, previsión meteorológica, localización,…) resultan interesantes para el usuario final que ayudarían a mejorar la experiencia de usuario y aportarían nuevas formas de coordinación, búsqueda,… al paradigma de la movilidad.
7. El **análisis de mecanismos para una interacción natural** y su posible inclusión como componentes del *middleware framework* es otra línea a considerar. La utilización de la voz para la creación de reglas o la utilización de interfaces sencillas y fáciles de usar (por ejemplo, definir macros de comportamiento mediante multi-toque de dispositivos NFC) puede mejorar la experiencia de interacción con el *middleware framework* desarrollado tal y como se ha recogido en el *feedback* del cuestionario realizado a los usuarios finales, siendo más amigable la interacción hombre-máquina que con el actual editor de reglas. La utilización de un personaje o agente virtual que pueda hablar, escuchar, ver, etc. que reaccione de manera sencilla e intuitiva, anticipándose a las necesidades de las personas, tal y como si fuera un asistente personal, puede ser otra desafiante área de trabajo a considerar en el futuro.

Bibliografía

[Abi05] ABI Research Forecasts over 100 Million Cellular / VoWi-Fi Phones in 2010. http://www.abiresearch.com/abiprdisplay.jsp?pressid=464

[Abi07] ABI Research: Social Communities Go Mobile: 174 Million Members Forecasted by 2011. http://www.tendencias21.net/Las-comunidades-virtuales-moviles-triplicaran-el-numero-de-sus-usuarios-en-2011_a1319.html

[Abowd+00] Abowd, G. D., Mynatt, E. D., “Charting Past, Present and Future Research in Ubiquitous Computing”, ACM Transactions on Computer-Human Interaction, Special issue on HCI in the new Millenium, 7(1):29-58, March 2000.

[Abrams+99] Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S.M., and Shuster, J. E. UIML: An Appliance-Independent XML User Interface Language, The Eighth International World Wide Web Conference, Toronto, Canada, May 11-14, 1999, http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html.

[Ali+02] Ali, M.F., Pérez-Quiñones, M.A., Abrams, M. and Shell E., Building Multi-Platform User Interfaces with UIML. in CADUI'2002, (France, 2002).[Allard+03] Allard J., Chinta V., Gundala S., Richard III G.. “Jini meets UPnP: An architecture for Jini/UPnP interoperability”. In The 2003 International Symposium on Applications and Design of User Interfaces.

[Allemang06] Allemang, D., Rule-based intelligence in the Semantic Web-or-"I'll settle for a web that's just not so dumb!", Second International Conference on Rules and Rule Markup Languages for the Semantic Web, Nov. 2006, ISBN 0-7695-2652-7, pages 83 – 88.

[Ballesteros+05] Ballesteros F.J., Leal K., Guardiola G., Soriano E.. “The Design and Implementation of Plan B 3rd edition. A dynamic distributed computing environment.” GSyC Tech. Rep 2004-05 Percom 2005.[Beer+07] Beer T., Rasinger J.,Höpken W., Fuchs M., Werthner H.. “Exploiting E-C-A Rules for Defining and Processing Context-Aware Push Messages”. LNCS volume 4824/2007, pages 199-206.

[Becker+03] Becker C., Schiele G., Gubbels H., Rothermel K., “BASE - A Micro-broker-based Middleware For Pervasive Computing”, Proceedings of

the 1st IEEE International Conference on Pervasive Computing and Communication, pp. 443-451, Fort Worth, USA, March 2003.

[Becker+04a] Becker C., Handte M., Schieke G., Rothermel K., “PCOM - Acomponent system for pervasive computing”, Conference on pervasive computing and communications, PERCOM 2004.

[Becker+04b] Becker C., Handte M., Schiele G., Rothermel K.. PCOM - a component system for pervasive computing. Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on, pages 67–76, 14-17 March 2004.

[Beckmann+03] Beckmann C. and Dey A., SiteView: Tangibly Programming Active Environments with Predictive Visualization, tech. report IRB-TR-03-019, Intel Research, 2003

[Beer+06] T. Beer, J. Rasinger, W. Höpken, M. Fuchs, H. Werthner. “Exploiting E-C-A Rules for Defining and Processing Context-Aware Push Messages”. LNCS volume 4824/2007, pages 199-206.

[Blackwell+01] Blackwell,A.F., Hague R., "AutoHAN: an architecture for programming the home," Human-Centric Computing Languages and Environments, 2001. Proceedings IEEE Symposia on Human Centric Computing Languages and Environments, pp. 150-157, 2001.

[BML10] Bean Markup Language, http://www.alphaworks.ibm.com/formula/bml, enero 2010.

[Bratko00] Bratko I. “Prolog Programming for Artificial Intelligence”. Longman Publishers, 3rd Edition, ISBN: 0201403757, August 2000.

[Brebner+04] Brebner P.,Ceccher E.,Marguerite J.,Tuma P., Ciuhandu O., Dufour B., Eeckhout L., Frenot S., Krishna A. S., Murphy J., Verbrugge C.. “Middleware benchmarking: approaches, results, experiences”. Concurrency and Computation: Practice and Experience. 2005; 17:1799-1805. DOI: 10.1002/cpe.918

[Chakravarthy+08] Chakravarthy S. and Adaikkalavan R. “Events and streams: Harnessing and unleashing their synergy!” In Proc. ACM Int. Conf. on Distributed Event-Based ystems, pages 1–12, 2008.

[Chen+05] Chen l., Li M., Cao J., Wang Y., “An ECA rule-based workflow design tool for Shangai grid”, Conference on services computing, SCC 2005.

[Cheverst+01] Cheverst, K., K. Mitchell, et al. (2001). Investigating Context-aware Informaiton Push vs. Information Pull to Tourists. MobileHCI’01 workshop on HCI with Mobile Devices, Lille, France.

[Cohen+99] Cohen J. and Aggarwal S. General Event Notification Architecture Base. Internet draft of the Internet Engineering Task Force (IETF), 1999. http://tools.ietf.org/html/draft-cohen-gena-p-base-01

[Conti+04] Conti M., Maselli G., Turi G., Giordano S., “Cross-Layering in Mobile Ad Hoc Network Design”, In IEEE Computer, Special Issue on Ad Hoc Networks, Feb. 2004.

[Continua10] Continua Health Alliance, http://www.continuaalliance.org/, enero 2010.

[Coulson+02] Coulson G., Blair G. S., Clarke M., Parlavantzas N.. “The design of a configurable and reconfigurable middleware platform”. Distributed Computing, 15(2):109--126, 2002

[DACIA10] DACIA Research proyect, http://www.eecs.umich.edu/~aprakash/dacia/publications.html, enero 2010.

[DaCosta+08] Da Costa C.A., Yamin A. C. and Geyer C.F.R.. “Toward a general software infrastructure for ubiquitous computing”. IEEE Pervasive Computing, Vol. 7, No. 1, pp. 64-73, Jan-March 2008.

[Dayu+05] Dayu Y., Peng Z., “event driven RFID reader for warehouse management” International conference on parallel and distributed computing, applications and technologies, PDCAT 2005.

[Dey01] Dey, A. “Understanding and Using Context,” J. Personal and Ubiquitous Computing, vol. 5, no. 1, pp. 4-7, Jan. 2001.

[Dey+01] Dey A. K., Salber D., Abowd G. D., “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications”, Anchor article of a special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), 2001, pp. 97-166.

[Dittrich+96] Dittrich K.R., Gatziu S. and Gepper A. “The Active Database Management System Manifesto: A Rulebase of ADBMS Features”. SIGMOD Record, September 1996

[Dongliang+08] Dongliang L., Kanyu Z., Xiaojing L., “ECA rule-based IO agent framework for greenhouse control system”, Symposium on computational intelligence and design, ISCID 2008.

[DLNA10] Digital Living Network Alliance, www.dlna.org, enero 2010.

[Eugster+03] Eugster P. T., Felber P. A., Guerraoui R., Kermarrec A-M., "The Many Faces of Publish/Subscribe", ACM Computing Surveys, Vol. 35, No. 2, June 2003, pp. 114–131.

[Esch04] Esch, J., “Computational Intelligence Methods For Rule-Based Data Understanding”, Proceedings of the IEEE, May 2004, Volume: 92, Issue:5, On page(s): 769 – 770.

[Fagor10] Fagor, http://www.fagor.com/es, enero 2010.

[Feigenbaum92] Feigenbaum, E. A. (1992). Expert Systems: Principles and Practice.

[Ferscha03] Ferscha A.. “Coordination in pervasive computing environments”. IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises”, pp. 3-9, Jun 2003.

[Forgy82] Forgy C.L. “Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem". Artificial Intelligence, vol. 19, no. 1, pp. 17-37, 1982.

[Friday+04] Friday A., Davies N., Wallbank N., Catterall E., and Pink S., “Supporting service discovery, querying and interaction in ubiquitous computing environments”, Wireless Networks., vol. 10, no. 6, pp. 631–641, 2004.

[Friedman-Hill09] Friedman-Hill E. , Jess: the Java Expert System Shell, Sandia Nat'l Laboratories, Albuquerque, N.M., 2001; http://herzberg.ca.sandia.gov/jess/

[G900] Toshiba company. http://www.toshiba-europe.com/mobile

[Garlan+02] Garlan D., Siewiorek D. P., Smailagic A., Steenkiste P., Project Aura: Distraction-Free Ubiquitous Computing, IEEE Pervasive Computing special issue on Integrated Pervasive Computing Environments 1, 2 (April-June 2002), 22-31.

[Grace+03] Grace P., Blair G. S., Samuel S.. “ReMMoC: A Reflective Middleware to Support Mobile Client Interoperability”. In Proceedings of International Symposium on Distributed Objects and Applications(DOA), Catania, Sicily, Italy, November 2003

[Gribble+00] Gribble S. D., Welsh M., Behren R., Brewer E. A., Culler D. E., Borisov N., Czerwinski S. E., Gummadi R., Hill J. R., Joseph A. D., Katz R. H., Mao Z. M., Ross S. and Zhao B. Y., “The Ninja architecture for robust Internet-scale systems and services”, Computer Networks. Special issue on Pervasive Computing 35, 4 (2000).the Internet (SAINT- 2003), Orlando, Florida (USA), January 2003.

[Guerrero+08] Guerrero P.E., and Sachs, K. “Performace evaluation of embedded eca rule engine”. LNCS volume 5261/2008, pages 48-63.

[HAVi10] Home Audio / Video Interoperability, http://www.havi.org/, enero 2010.

[Headon+02] Headon R., Curwen R., “Movement Awareness for Ubiquitous Game Control”, Personal and Ubiquitous Computing (2002) 6:407-415.

[Herbert+08] Herbert, J.; O'Donoghue, J.; Chen, X., "A Context-Sensitive Rule-Based Architecture for a Smart Building Environment," Future Generation Communication and Networking, 2008. FGCN '08. Second International Conference on , vol.2, no., pp.437-440, 13-15 Dec. 2008.

[Hicks07] Hicks R. C., The no inference engine theory – Performing conflict resolution during development, Decision support systems 2007 vol. 43, 435-444

[Hightower +01] Hightower J., Borriello G.. “Location systems for ubiquitous computing”. IEEE Computer, 34(8):57-66, August 2001.

[Hollink+05] Hollink L., Little S., Hunter J., “Evaluating the Application of Semantic Inferencing Rules to Image Annotation”, Proceedings of the 3rd international conference on Knowledge capture, pages: 91 – 98, 2005

[Hooper99] Hopper A. “Sentient Computing”, The Royal Society Clifford Paterson Lecture, AT&T Laboratories Cambridge, Technical Report 1999.12, 1999

[Horrocks+04] Horrocks I., Patel-Schneider P. F., Boley H., Tabet S., Grosof B.,Dean M.. “Swrl: A semantic web rule language combining owl and ruleml”. Technical Report W3C Member Submission - 21, 2004.

[IEC10] International Engineering Consortium, http://www.iec.org/, enero 2010.

[IEEE Glossary90] Institute of Electrical and Electronics Engineers. IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries, 1990.

[IEEE00] “IEEE 100 The Authoritative Dictionary of IEEE Standards Terms Seventh Edition”, IEEE Press, ISBN 0-7381-2601-2

[IEEE91] IEEE, Ed. (1991). IEEE Standard Computer Dictionary - Compilation of IEEE Standard Computer Glossaries. New York, IEEE.

[JavaBeans10] JavaBeans technology, java.sun.com/products/javabeans, enero 2010.

[Jini10] Jini Foundation, http://www.jini.org/, enero 2010.

[Jung+07] Jung J-Y.,Park J., Han S-K.,Lee K., “An ECA-based framework for decentralized coordination of ubiquitous web services”, Information and software technology 49 (2007) 1141-1161.

[Kendall+99] Kendall, J. E. and K. E. Kendall (1999). Information Delivery Systems: An Exploration of Web Pull and Push Technologies. Communications

of the AIS 1(4).[Kozuch+02] Kozuch M., Satyanarayanan M., “Internet Suspend/Resume”, Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA’02), IEEE 2002.

[LDAP] LDAP (Lightweight Directory Access Protocol): http://www.gracion.com/server/whatldap.html

[Lee05] Lee W. “Personalizing Information Services for Mobile Users”, Soft Computing: Methodologies and Applications, 2005.

[Lee+07] Lee H., Park J., Park P,. Jung M., Shing D..”Cynamic conflict detection and resolution in a human-centered ubiquitous environment”, Universal access in HCI, HCII 2007, LNCS pp. 132-140, 2007.

[Li+04] Li Y., Hong J. I., Landay J. A. Topiary: a tool for prototyping location-enhanced applications. In UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology. ACM Press, New York, NY, USA, pp. 217-226, 2004.

[Li+09] Li X.,Jia Z.,Lu X.,Wang H.. “Rule-based publish-subscribe mechanism for real-time applications”, IEEE/ACIS International conferece on computer and information science 2009.

[Liman+05] Limam N., Ziembicki J., Ahmed R., Iraqi Y., Li T., Boutaba R., Cuervo F., “OSDA: Open Service Discovery Architecture for Efficient Cross-domain Service Provisioning”, Computer Communications Journal, Special Issue on Emerging Middleware for Next Generation Networks. 2005.

[López-de-Ipiña01] López-de-Ipiña D. “An ECA Rule-Matching Service for Simpler Development of Reactive Applications”. Middleware 2001, Heidelberg, Germany, 12-16 November 2001.

[López-de-Ipiña+06] López-de-Ipiña D., Vázquez J.I., García D., Fernández J., García I., Sainz D., Almeida A., “EMI2lets: a Reflective Framework for Enabling AmI”, Journal of Universal Computer Science (J.UCS), vol. 12, no. 3, pp. 297-314, March 2006.

[Mandarax10] The Mandarax Project, http://mandarax.sourceforge.net/, enero 2010.

[McBurney+07] McBurney S., Williams M.H., Taylor N. K., Papadopoulou E.,”Managing user preferences for personalization in pervasive service environment”, Advanced international conference on telecommunications, AICT 2007.

[MICAz] MICAz de Crossbow Technology:http://www.xbow.com/ Home/wHomePage.aspx

[Mozilla10] Comunidad Mozilla. www.mozilla.org, enero 2010.

[NASA99] NASA, “CLIPS: A Tool for Building Expert Systems”. http://www.ghg.net/clips/CLIPS.html, August 99.

[Nishigaki+05] Nishigaki, K., Yasumoto, K., Shibata, N., Ito, M., and Higashino, T. 2005. Framework and Rule-Based Language for Facilitating Context-Aware Computing Using Information Appliances. In Proceedings of the First international Workshop on Services and infrastructure For the Ubiquitous and Mobile internet (Siumi) (Icdcsw'05) - Volume 03 (June 06 - 10, 2005).

[Olmedo-Aguirre+08] J. O. Olmedo-Aguirre, M. Rivera de la Rosa, G. Morales-Luna: ECA-Rule Visual Programming for Ubiquitous and Nomadic Computing. MICAI 2008: 925-935.

[OPC10] OPC Foundation, http://www.opcfoundation.org/, enero 2010.

[OpenLaszlo10] Open source platform for the development and delivery of rich Internet applications. http://www.openlaszlo.net/, enero 2010.

[OpenLDAP10] Lightweight Directory Access Protocol Software, OpenLDAP Foundation, www.openldap.org, enero 2010.

[Oscar+07] Pérez O., Piccardi M., García J., Molina J.M., “Comparison of Classifiers for Human Activity Recognition”, J. Mira and J.R. Alvarez (Eds.): IWINAC 2007, Part II, LNCS 4528, pp. 192–201, 2007.

[OSGI10] OSGI Alliance, http://www.osgi.org/, enero 2010.

[Papadopoulou+08] Papadopoulou E., McBurney S., Taylor N., Williams M.H., “Linking privacy and user preferences in the identity management ofr a pervasive system”, International conference on web intelligence and intelligent agent technology, WIIAT 2008.

[Parra+09] Parra J., Hossain M.A., Uribarren A., Jacob E., El Saddik A,. “Flexible Smart Home Architecture using Device Profile for Web Services: a Peer-to-Peer Approach”. International Journal of Smart Home Vol.3, No.2, April, 2009.

[Paton+99] Paton N.W. and Díaz O. “Active Databases Survey”. ACM Computing Surveys, vol.31, no.1, pp. 63-103, March 1999.

[PerCom10] Annual Conference on Pervasive Computing and Communications, www.percom.org. Enero 2010.

[Perez+07] Perez O., Piccardi M., García J., and Molina J.M.. “Comparison of Classifiers for Human Activity Recognition”, IWINAC 2007, Part II, LNCS 4528, pp. 192–201, 2007.

[Pietzuch+03] P. Pietzuch, B. Shand, J. Bacong. “A Framework for Event Composition in Distributed Systems”. ACM/IFIP/USENIX Int. Middleware Conference 2003. Springer-Verlag.

[Polleres+08] A. Polleres, H. Boley, and M. Kifer. Rif datatypes and built-ins 1.0, 30 July 2008.

[Powers00] Powers S., Digital Play Dough. Designing Applications with XUL, WebTechniques, vol. 5, issue 7, (July 2000).

[Pronto09] Home automation controllers and remotes manufactured and supported by Phlips. http://www.pronto.philips.com/, octubre 2009.

[Ravi+05] Ravi N,. Dandekar N,. Mysore P,. Littman M. L., Activity Recognition from Accelerometer Data, American Association for Artificial Intelligence 2005.

[ReflectionJAVA10] The JAVA Tutorials. http://java.sun.com/docs/books/tutorial/reflect/, enero 2010.

[ReflectionNET10] Reflection Overview. http://msdn.microsoft.com/en-us/library/f7ykdhsy(VS.71).aspx, enero 2010.

[Ridong+09] Ridong J., Kee T.Y.,Yuen W.C.,Limbu D.K., “Development of event-driven dialogue system for social mobile robot”, Global Congress on Intelligent Systems, GCIS 2009.

[Roman+02] Roman M., Hess C. K., Cerqueira R., Narhstedt K., Campbell R. H., “Gaia: A middleware infrastructure to enable active spaces”, Technical Report UIUCDCSR20022265. University of Illinois at Urbana Champaign, 2002.

[RuleML09] RuleML language: http://ruleml.org/ (accedida en 2009).

[Saha+03] Saha D. and Mukherjee A.. “Pervasive computing: a paradigm for the 21st century”. Computer Vol. 36(3), pp. 25-31, March 2003.

[Salutation05] Salutation Consortium, www.salutation.org, Abril 2005.

[Salvador+05] Salvador Z., Lafuente A., Larrea M.. “Jini as a platform for ubiquitous computing” <http://www.sc.ehu.es/acwlaalm/research/jini-ucami-2005.pdf>, in Proceedings of the Simposio sobre Computación Ubicua e Inteligencia Ambiental, UCAmI 2005, pp. 251-257, Granada, Spain, Sep 2005.

[Sameh+04] Sameh A., El-Kharboutly R., “Modeling Jini-UPnP Bridge using Rapide ADL” Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS'04), Beirut, Lebanon, July 2004, p. 237.

[Sashima+04] Sashima A., Izumi N., Kurumatani K., “Location-mediated web services coordination in ubiquitous computing”, International Conference on Services Computing, SCC 2004.

[Satyanarayanan01] Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. IEEE Personal Communications, 8(4):10-17.

[Satyanarayanan04] Satyanarayanan M., Carnegie Mellon University & Intel Research Pittsburgh, “Seamless Mobility: in pursuit of the Holy Grail”, PERCOM 2004 KeyNote Speech

[Sawhney+00] Sawhney N., Schmandt C., “Nomadic Radio: Speech and Audio Interaction for Contextual Messaging in Nomadic Environments,” ACM Trans. on Computer Human Interaction, vol. 7, no. 3, Sept. 2000, pp. 353–383. [Weiser91] Weiser M., “The Computer for the Twenty-First Century”, Scientific American, pp. 94-104, September 1991.

[Schiefer+07] Schiefer, J., Rozsnyai, S., Rauscher, C., and Saurer, G. 2007. Event-driven rules for sensing and responding to business situations. In Proceedings of the 2007 inaugural international Conference on Distributed Event-Based Systems (Toronto, Ontario, Canada, June 20 - 22, 2007). DEBS '07, vol. 233. ACM, New York, NY, 198-205.

[Sean-Woo+02] Seon-Woo Lee Mase, K, “Activity and location recognition using wearable sensors”, Pervasive Computing, IEEE2002 Volume: 1, Issue: 3On page(s): 24- 32

[Shankar+05] Shankar, C.S.; Ranganathan, A.; and Campbell, R., "An ECA-P policy-based framework for managing ubiquitous computing environments," Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on, vol., no., pp. 33-42, 17-21 July 2005.

[Shankar+06] Shankar, C.S.,Campbell, R.,”Ordering management actions in pervasive systems using specification-enhanced policies”, Conference on pervasive computing and communications, PERCOM 2006.

[SLP10] Open SLP, http://www.openslp.org/, enero 2010.

[SmartHome09] SMARTHOME 2009: http://www.smarthome.com/_/index.aspx

[Stefanelli+08] Stefanelli C., Tortonesi M., “Session Mobility in the Mockets Communication Middleware”, 2008 IEEE.

[Thomson+07] Thomson G., Sacchetti D., Bromberg Y., Parra J., Georgantas N., and Issarny V. “Amigo Interoperability Framework: Dynamically Integrating Heterogeneous Devices and Services”, Constructing Ambient Intelligence AmI 2007 Workshops, Darmstadt, Germany, November 2007.

[UbiComp10] ACM International Conference on Ubiquitous Computing. http://www.ubicomp2010.org/, enero 2010.

[UPnP10] UPnP Forum. Universal Plug and PlayTM Device Architecture, July 2008. http://www.upnp.org/, enero 2010.

[UPnP-desc10] Standardized device and service descriptions: http://www.upnp.org/standardizeddcps/default.asp, enero 2010.

[Vastenburg+07] Vastenburg M. H., Keyson D. V. and Ridder H. “Measuring User Experiences of Prototypical Autonomous Products in a Simulated Home Environment”, HCI (2): 998-1007, 2007.

[VMware10] Virtualización para escritorios y servidores x86. Http://www.vmware.com/. Enero 2010.

[W3C10] The World Wide Web Consortium (W3C), http://www.w3.org/, enero 2010.

[Walzer+08] Walzer, K., Breddin, T., and Groch, M. 2008. Relative temporal constraints in the Rete algorithm for complex event detection. In Proceedings of the Second international Conference on Distributed Event-Based Systems (Rome, Italy, July 01 - 04, 2008). DEBS '08, vol. 332. ACM, New York, NY, 147-155, 2008.

[Weis+06] Weis T.,Handte M., Knoll M., Becker C., “Customizable pervasive applications”, Conference on pervasive computing and communications, PERCOM 2006

[Weis+07] Weis, T., Knoll, M., Ulbrich, A., Muhl, G., and Brandle, A. 2007. Rapid Prototyping for Pervasive Applications. IEEE Pervasive Computing 6, 2 (Apr. 2007), 76-84.

[Weiser91] Weiser M., “The computer of the 21st century”, Scientic American, vol. 265, no. 3, pp. 66–75, Sept. 1991.

[Whiteboard04] Whiteboard pattern. Technical Whitepaper, august 2004 . http://www.osgi.org/wiki/uploads/Links/whiteboard.pdf

[WM6] Microsoft Windows Mobile. http://www.microsoft.com/windowsmobile/en-us/default.mspx

[WS-Eventing06] Web Service Eventing. http://www.w3.org/Submission/WS-Eventing/, marzo 2006.

[XAML10] Extensible Application Markup Language (XAML), http://msdn.microsoft.com/en-us/library/ms747122.aspx, enero 2010.

[Zhang+04] Zhang, T. and Brügge, B.: Empowering the User to Build Smart Home Applications, ICOST 2004.

[Zhang+05] Zhang C., Li M., Pan Q., “An ECA rules based middleware architecture for wireless sensor networks”, Parallel and distributed computing, applications and technologies, PDCAT05.

Anexo A: Dispositivos virtuales UPnP

En este anexo se describe cómo crear dispositivos virtuales y localizarlos en el hogar, en una oficina o en cualquier otro lugar, con el objetivo de crear un entorno virtual que nos permita estudiar y validar el funcionamiento del editor y del motor de reglas.

A.1 Introducción

Un dispositivo virtual puede estar compuesto por uno o varios servicios que intentan emular la funcionalidad ofrecida por un dispositivo físico cualquiera en un entorno. En general, un servicio de un dispositivo expone una lista de acciones a las que el servicio responderá y una lista de variables que muestran el estado del servicio. En nuestro caso, se hará uso de la arquitectura UPnP (Universal Plug & Play) que permite el intercambio de información y datos entre los dispositivos conectados en una red [UPnP10]. Esta arquitectura permite el anuncio, descubrimiento, envío/recepción de eventos y control remoto de los dispositivos en red. En nuestro caso, los dispositivos virtuales serán dispositivos UPnP.

Para facilitar la creación de los dispositivos virtuales se ha creado un editor. Este editor permite la creación de dispositivos virtuales con sus servicios y acciones correspondientes, así como la localización de los mismos en una estancia de una casa, oficina, etc. A continuación, se dará una descripción de la interfaz del editor y se mostrará su uso.

A.2 Creación de dispositivos virtuales

En esta sección, en primer lugar, se describirá la especificación de UPnP necesaria para definir un dispositivo UPnP, y se mostrará un punto de control genérico que permite controlar los dispositivos virtuales creados. Aunque será el Motor de Reglas (MR) quien controle los valores de las variables e invoque acciones, este punto de control resulta de gran utilidad para estudiar el funcionamiento general del sistema. Concretamente se utilizará para visualizar y controlar los dispositivos virtuales UPnP existentes en la red local mientras se realizan las configuraciones en la validación. En segundo lugar, se describirá el editor que facilita la creación de los dispositivos virtuales, y se mostrará su funcionamiento a través de un ejemplo.

A.2.1 Especificación de UPNP

Dado que los dispositivos creados van a ser dispositivos UPnP, la arquitectura de los mismos se ha definido de tal forma que cumple con la especificación de UPnP. Acorde con dicha especificación, un dispositivo UPnP puede contener otros dispositivos

embebidos (embarcados o lógicos) y servicios. Cada dispositivo embebido a su vez estará formado por uno o más servicios. Cada servicio está compuesto por una lista de acciones y una lista de variables. En las acciones también se definen los parámetros o argumentos de cada acción. Las variables, por otra parte, muestran el estado de los servicios.

Esta especificación de UPnP agrupa en un mismo dispositivo varias funcionalidades. La misma configuración siguen por ejemplo, las motas (motes, en inglés) MICAz de Crossbow [MICAz], en las que, dependiendo de la funcionalidad requerida (p.ej., temperatura, presión, acelerómetro, comunicaciones *bluetooth, zigbee, WiFi*), se añaden las placas de sensores correspondientes. Esto es, el mismo dispositivo puede ofrecer funcionalidades diferentes dependiendo del hardware del que esté compuesto en cada momento.

Por otra parte, la propia especificación UPnP define unas categorías de servicios a tener en cuenta al definir la lista de servicios de los dispositivos. Sin embargo, esta lista de categorías de servicios estandarizados es limitada (p.ej. audio/vídeo, luz, impresora, *gateway, router WLAN*, automatización del hogar) [UPnP10]. Así pues, para poder cubrir el rango de servicios que podrían estar inmersos en un entorno del hogar, se ha extendido esta lista de servicios para que incluya dispositivos tan básicos como los ofrecidos por los sensores de presencia, actuadores, potenciómetros u otros que se describirán a continuación.

A.2.2 Punto de control de los dispositivos UPNP

Como se explica en el trabajo de tesis, toda variable o consigna (variable deseada) de un dispositivo puede ser modificada localmente y remotamente. Localmente será a través de la interfaz de cada dispositivo virtual, mientras que remotamente puede ser a través de un punto de control genérico de los dispositivos UPnP o bien a través del propio MR que se encargará de cambiar y ejecutar las acciones correspondientes. La figura A.1 muestra el punto de control genérico de Siemens.

Este punto de control permite ver todos los dispositivos UPnP de la red, ver sus servicios y sus acciones, invocar dichas acciones y cambiar las variables de los dispositivos. La figura A.1 muestra las acciones y variables asociadas al servicio de información. A través de las acciones del tipo set_... se pueden cambiar los valores de la consigna, y a través de las acciones del tipo get_... se pueden obtener los valores actuales de las variables.

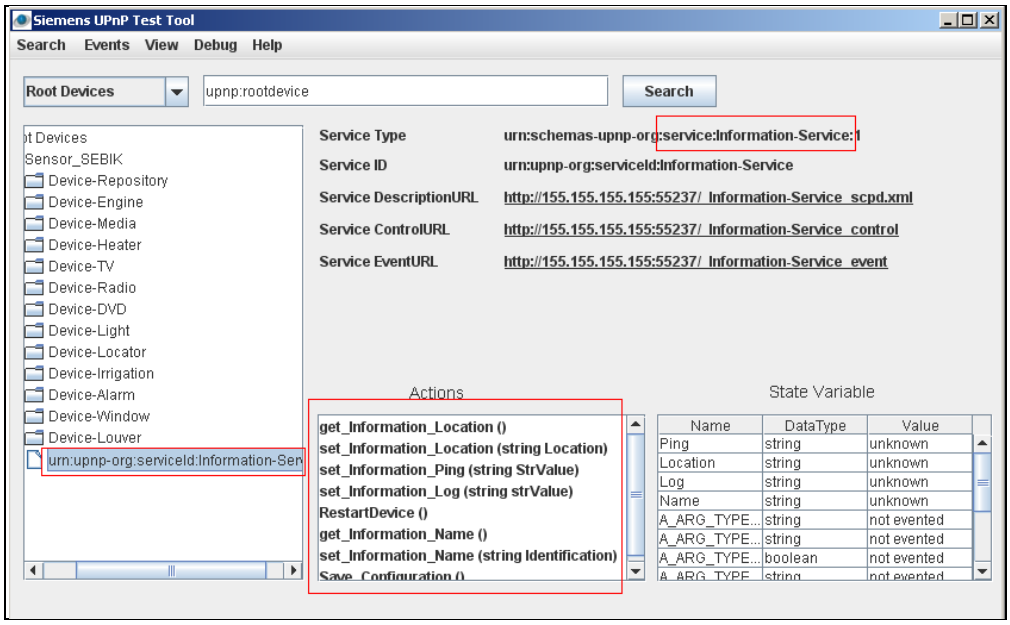


Fig. A.1 Vista de las acciones y variables del servicio *Information* en el punto de control genérico de Siemens

A.2.3 Descripción del editor

Con el objetivo de facilitar la comprensión, cuando se crea un dispositivo a través del editor, todos los servicios posibles son mostrados al usuario. De esta manera, el usuario conoce todos los servicios existentes y puede seleccionar aquellos que desee simplemente activándolos o desactivándolos. Por defecto, todos estarán desactivados.

Una vez se haya creado un dispositivo y sus servicios, estos aparecerán en una columna a la izquierda siguiendo el diseño general del 'Explorador de las carpetas de Windows', con el que haciendo clic sobre el signo más o el menos se expande o contrae la información. La figura A.2 muestra la información del dispositivo 'Sensor_SEBIK' que contiene 13 dispositivos embebidos y un servicio 'Information_Service'. Este servicio contiene ocho acciones y es común a todos los dispositivos. Sirve para obtener información sobre el dispositivo.

El dispositivo generado es una aplicación de MS .NET, que incluye una interfaz gráfica a través de la cual un usuario puede cambiar los valores de las variables, invocar servicios o utilizar dispositivos embebidos del dispositivo (funcionalidad del dispositivo). Por otro lado, al crear un dispositivo, se implementa una interfaz UPnP a través de la que el dispositivo expone sus servicios al resto de los dispositivos de la red. Así pues, como se ha explicado anteriormente, cuando un usuario modifica alguna variable de estado del dispositivo usando la interfaz, los cambios son comunicados por medio de la infraestructura UPnP al resto de dispositivos. A la inversa, si algún cambio es realizado a través de la interfaz UPnP, la variable afectada del dispositivo es actualizada en la interfaz gráfica. Además, cualquier cambio en el valor de una variable, por el mecanismo de *eventing* de la interfaz UPnP, es enviado a todos los dispositivos u aplicaciones interesados.

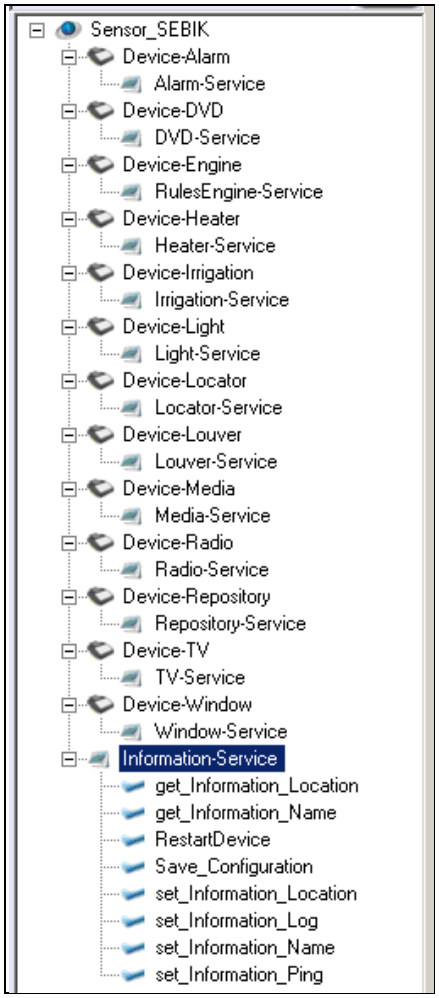


Fig. A.2 Vista de un dispositivo UPnP (Sensor_SEBIK), sus dispositivos embebidos y los servicios de los mismos

Finalmente, si queremos emular la existencia de cualquier servicio en una estancia del hogar, no tendremos más que ejecutar (instanciar) tantas veces como se requiera el simulador, activar el servicio deseado e indicar en la descripción su ubicación. A partir de ese momento, dispondremos de una serie de dispositivos ubicados en toda la casa para su control.

A.2.4 Ejemplo de creación de un dispositivo

A continuación se describirán los servicios activables y disponibles para la configuración del dispositivo virtual.

Servicio de información

El servicio 'Information' o servicio de información es un servicio que se encuentra en todos los dispositivos UPnP (Fig. A.3). Todos y cada uno de los dispositivos contendrán este servicio. Este servicio ofrece información de tipo genérica del dispositivo, como es el nombre del dispositivo, su ubicación, etc.

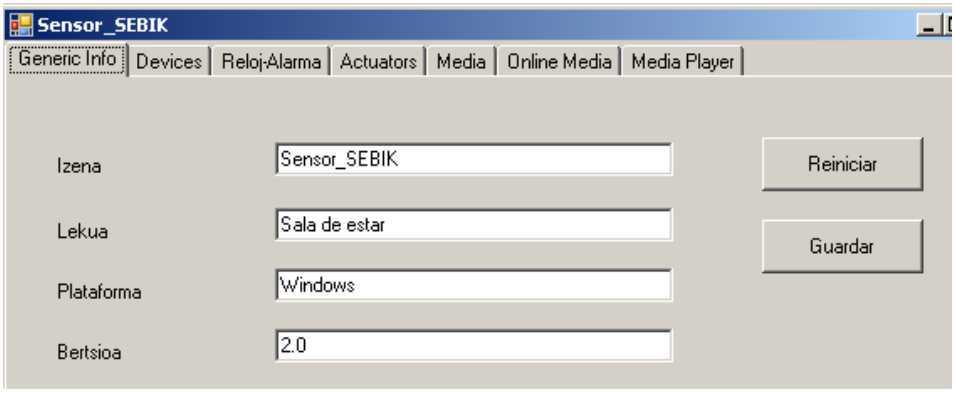


Fig. A.3 Información genérica del dispositivo virtual

Servicio de calefacción o enfriamiento

El servicio calefacción o 'Heater' pretende emular un servicio de un dispositivo de tipo calefacción o aire acondicionado (frío/calor), de tal manera que, dada una consigna (variable deseada), el dispositivo modificará la temperatura hasta lograr la consigna en intervalos de medio grado cada segundo. La figura A.4 muestra que partiendo del valor de la temperatura actual y en modo auto el dispositivo irá incrementando dicho valor hasta alcanzar el valor de consigna marcado. Con este mecanismo, se pretende emular el funcionamiento del calentamiento/enfriamiento de una estancia, que se produce poco a poco en el tiempo.

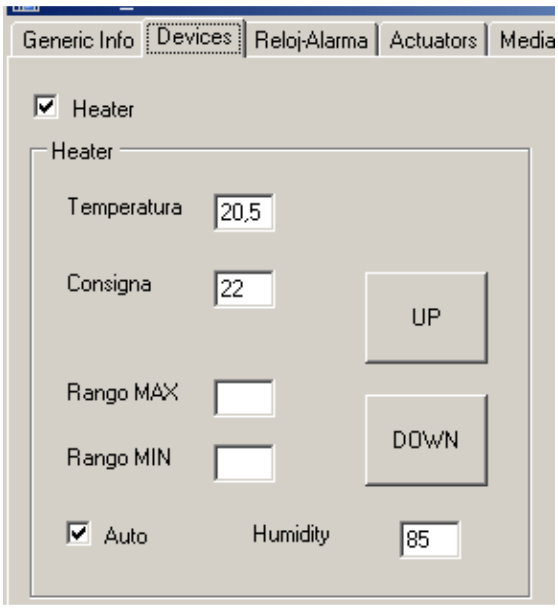


Fig. A.4 Servicio Heater o calefacción-enfriamiento

La figura A.5 muestra las acciones y variables del servicio Heater en el punto de control genérico de Siemens. Por ejemplo, *get_Heater_Temperature*() permite obtener la temperatura que hay actualmente o *set_Heater_State*(ON) enciende (simula el encendido) el aparato en cuestión.

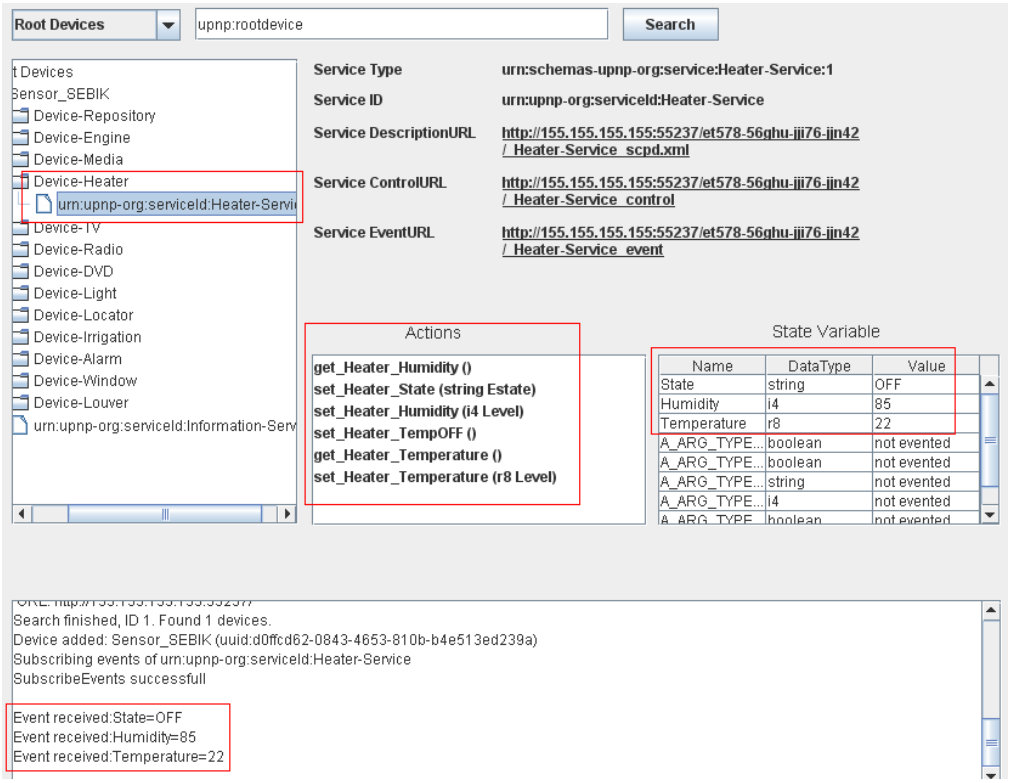


Fig. A.5 Vista de las acciones y variables del servicio Heater en el punto de control genérico de Siemens

Servicio luz

El servicio 'Light' o luz pretende emular un dispositivo de tipo bombilla o lámpara que permiten regular la luz (Fig. A.6). Se ha considerado que la intensidad de la luz pueda tomar los valores del 0 al 10, donde el cero significa que la bombilla o lámpara está apagada y el valor diez indica el encendido con la intensidad máxima.

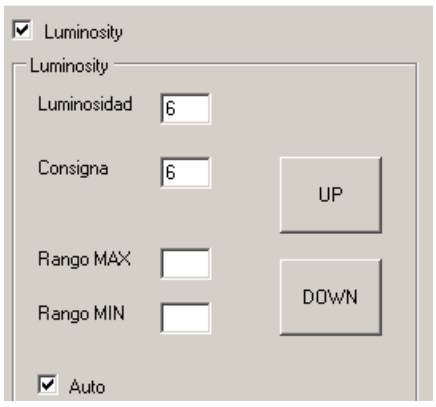


Fig. A.6 Vista dispositivo Luz

Como anteriormente se ha descrito, toda consigna puede ser modificada remotamente y, en cada cambio de valor, una notificación es enviada a los dispositivos interesados. En la parte inferior de la figura A.7 se muestra la suscripción de este servicio y la recepción automática del valor de la variable 'Light' cuando cambia su valor.

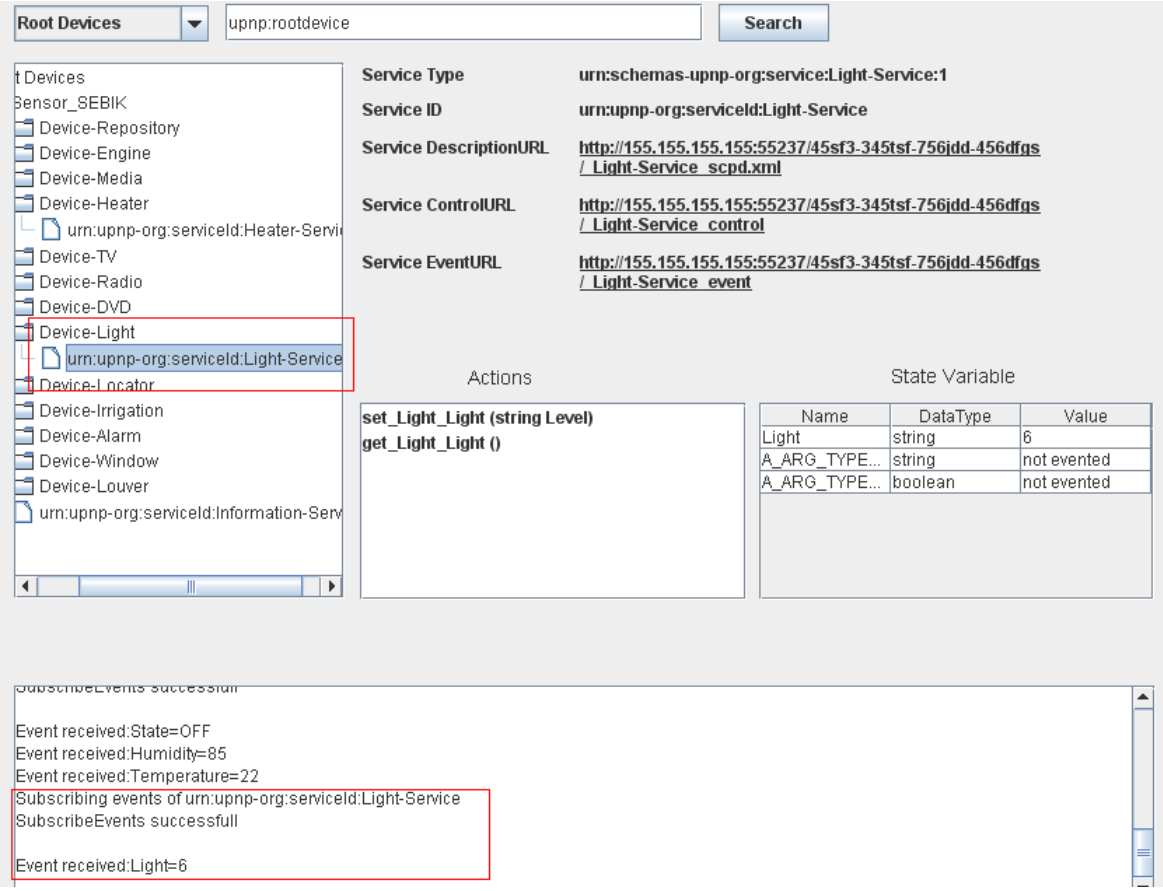


Fig. A.7 Vista de las acciones y variables del servicio luz en el punto de control genérico de Siemens

Servicio localizador

El servicio 'Locator' o localizador del usuario pretende simular un sensor/detector de presencia (Fig. A.8), de tal manera que si introducimos un valor con el nombre o identificador de una persona, emularemos que dicha persona o dispositivo se encuentra en la ubicación prefijada.

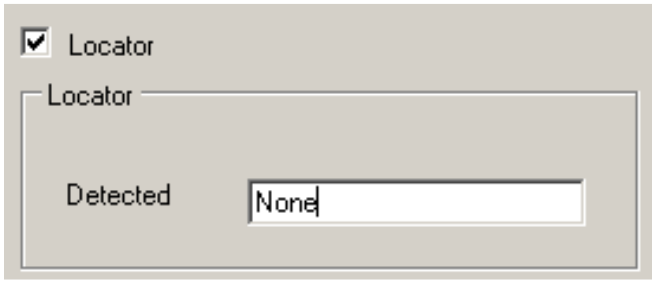


Fig. A.8 Vista variable localizador

Cada vez que la información del servicio de localización es modificada, un evento es enviado a la red para informar del cambio a todas las aplicaciones o dispositivos interesados. La figura A.9 muestra los efectos de la suscripción al servicio de localizador, recibiendo un mensaje con el valor actual de la variable 'User'.

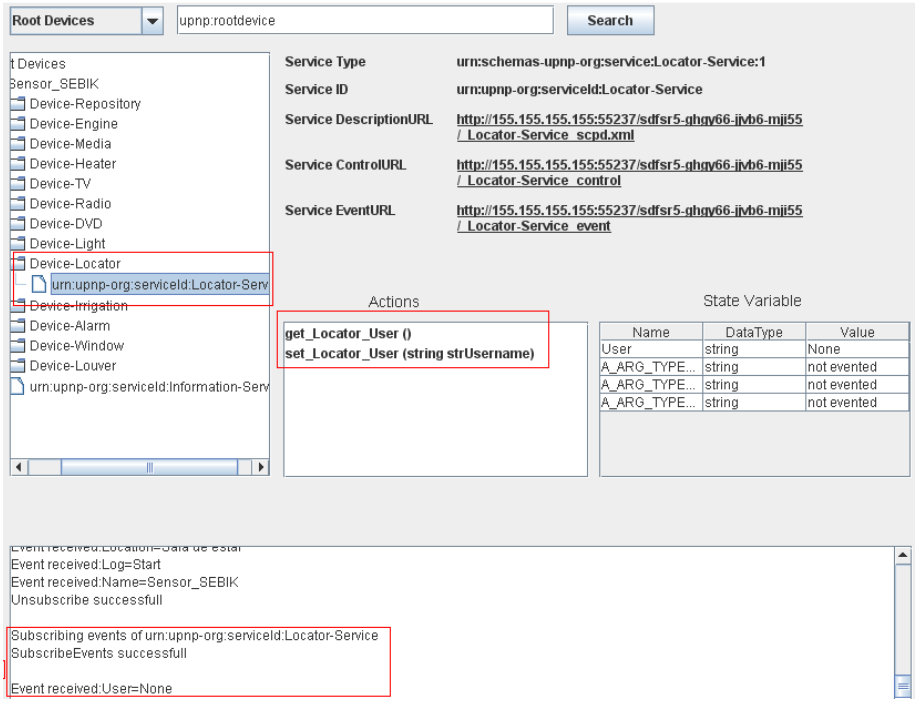


Fig. A.9 Vista de las acciones y variables del servicio localizador. Punto de control genérico Siemens

Servicio Reloj-Alarma

El servicio 'Alarm-Clock' o reloj-alarma pretende simular el funcionamiento de un despertador (Fig. A.10). El usuario tanto local como remotamente puede definir alarmas para que ocurran a lo largo del tiempo.

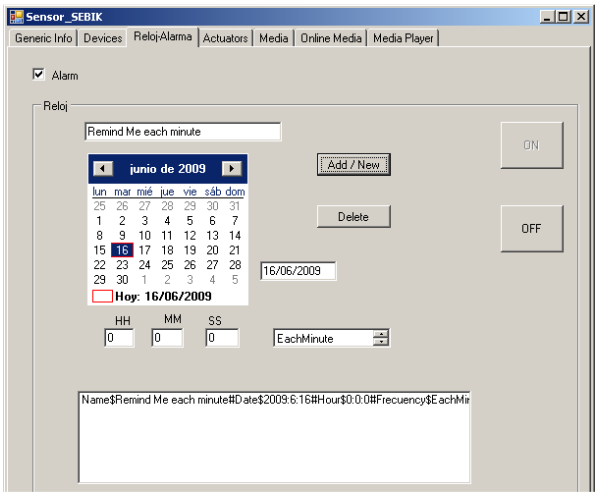


Fig. A.10 Vista variables Alarm-Clock

Las alarmas pueden ser incluso periódicas en el tiempo, de tal forma que cuando se cumpla la fecha de la alarma, un evento es enviado a la red para su procesamiento. La figura A.11 muestra la información que ofrece el servicio alarma, con sus variables, acciones y eventos que se reciben después de realizar la suscripción.

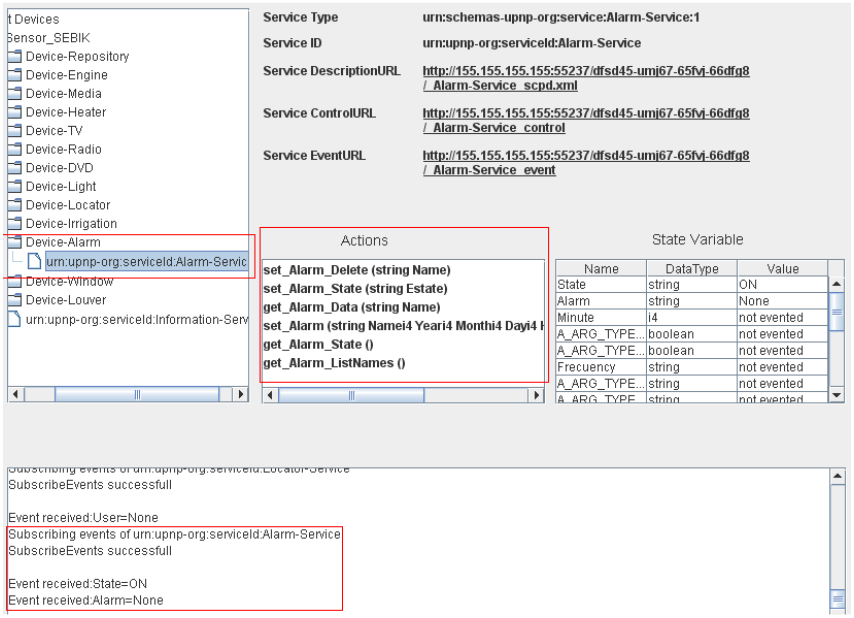


Fig. A.11 Vista de las acciones y variables del servicio reloj-alarma en el punto de control de Siemens

Tal y como se puede apreciar en la figura A.12, el mensaje enviado por el servicio reloj-alarma como evento del cumplimiento de una alarma contiene toda la información. En este caso, si cortamos el mensaje cada vez que aparece el símbolo "#", obtendremos las parejas de variables con sus valores. Por ejemplo, la pareja de variable "Name$Remind Me each minute" nos indica que la descripción de la alarma es 'minutal', es decir, cada minuto se va a producir el evento. Por otro lado, la pareja de variable "Date$2009:6:16" nos indica que el evento se ha producido un 16 de junio de 2009. Y así con el resto de variables. El mensaje anteriormente descrito (información del evento) será recibido por los dispositivos o programas que se hayan suscrito al servicio de reloj-alarma. En caso de necesitar información suplementaria, se utiliza la función 'get_AlarmData ()', obteniendo la información completa relativa a la alarma definida y en el formato predefinido.

Fig. A.12 Valor de información de retorno de una alarma definida (get_AlarmData).

Servicios de varios actuadores: persianas, ventanas y válvulas

Para simular los dispositivos actuadores, se han creado tres servicios:

a) Servicio Louver: para apertura y cierre de las persianas.
b) Servicio Irrigation: para la apertura y cierre de una válvula o grifo.
c) Servicio Window: para apertura y cierre de las ventanas.

Todos los servicios se han agrupado en una misma ventana (Fig. A.13).

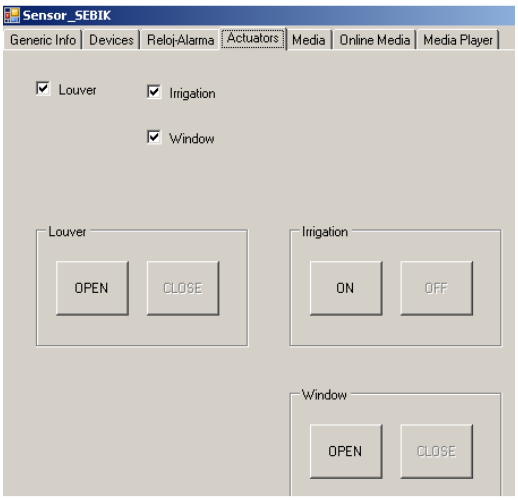


Fig. A.13 Vista de los servicios y variables de varios actuadores

Los valores de las variables de los actuadores van a ser, 'ON/OFF' en el caso de la válvula, y 'OPEN/CLOSE' en el caso de las ventanas y persianas. La figura A.14 muestra las acciones y variables del servicio Window en el punto de control genérico de Siemens.

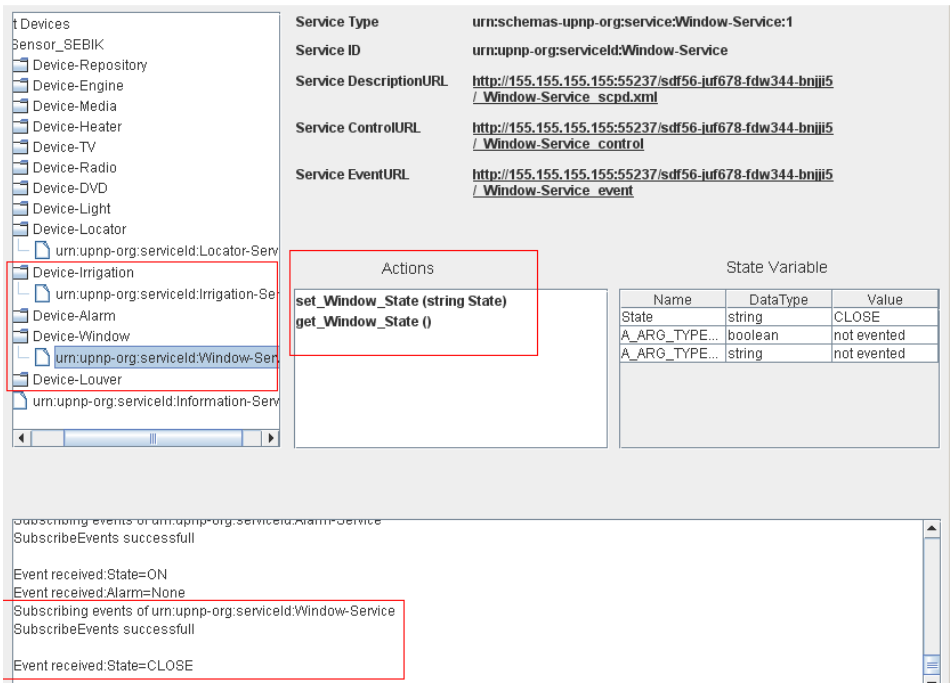


Fig. A.14 Vista de las acciones y variables del servicio Window en el punto de control genérico de Siemens

Servicio Multimedia

Para simular dispositivos multimedia, se han creado tres servicios:

1) Servicio TV: para simular un televisor.
2) Servicio Radio: para simular una radio.
3) Servicio DVD: para simular un reproductor de DVD.

Debido a su funcionalidad común, se han agrupado en una única interfaz de usuario (Fig. A.15). En ella se muestran las funciones principales que este tipo de aparatos ofrecen, desde la puesta en marcha 'ON/OFF' hasta la posibilidad de seleccionar el canal deseado y el volumen.

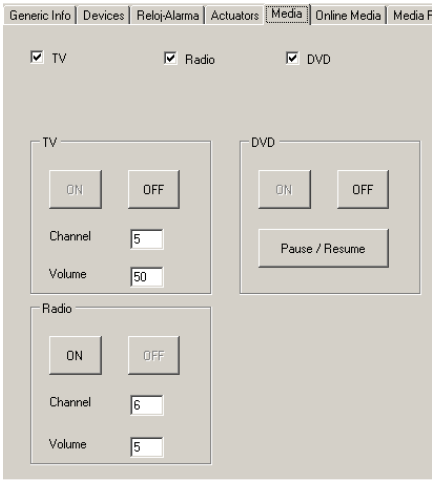


Fig. A.15 Vista servicios Multimedia

Servicio Live-Media

Con el servicio 'Live-Media' se pretende simular la existencia de altavoces y pantallas adicionales a través de los que se puedan visualizar mensajes de notificación, ver películas o ver la televisión (Fig. A.16).

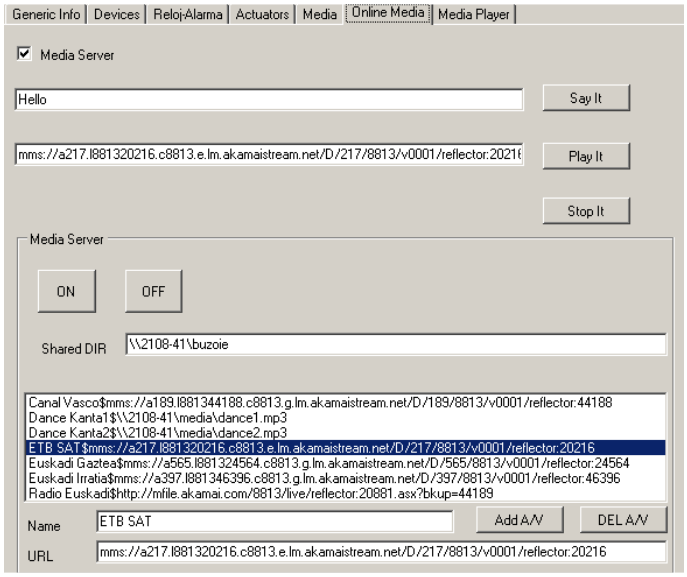


Fig. A.16 Vista de los servicios Live-Media

Se trata de un dispositivo que tiene la capacidad de conectarse a Internet y reproducir radio y televisión que se emite on-line. Se diferencia de la televisión que se ha descrito en el servicio Multimedia en que para este caso se debe indicar una dirección URL, tal y como se muestra en la figura A.16.

Por otro lado, también disponemos de un servicio de síntesis de texto (*Text-To-Speech - TTS*). De tal manera que remotamente se pueden enviar mensajes para su locución y notificación hacia el usuario. Se ha utilizado la tecnología de MS Speech para realizar esta tarea. Por ejemplo, si se desea que cada vez que entremos en casa, nada más pasar la puerta alguien nos diga “Hola, ¿qué tal?” utilizaremos este mecanismo del dispositivo Live-Media.

El servicio de Live-Media se extiende con repositorios remotos donde se alberga contenido multimedia (ficheros de tipo media, .avi, .mpg, .mp3, .wav…), con lo que conociendo una ubicación remota donde se almacenan ficheros de este tipo, el usuario podrá seleccionar el contenido deseado y será capaz de reproducirlo utilizando el reproductor por defecto de Windows Media en la ubicación requerida (Fig. A.17). La utilización de Windows Media obedece a razones de practicidad y comodidad, ya que se integra totalmente y es fácilmente utilizable desde el entorno de desarrollo .NET que hemos seguido. No habría inconvenientes en utilizar otros reproductores.

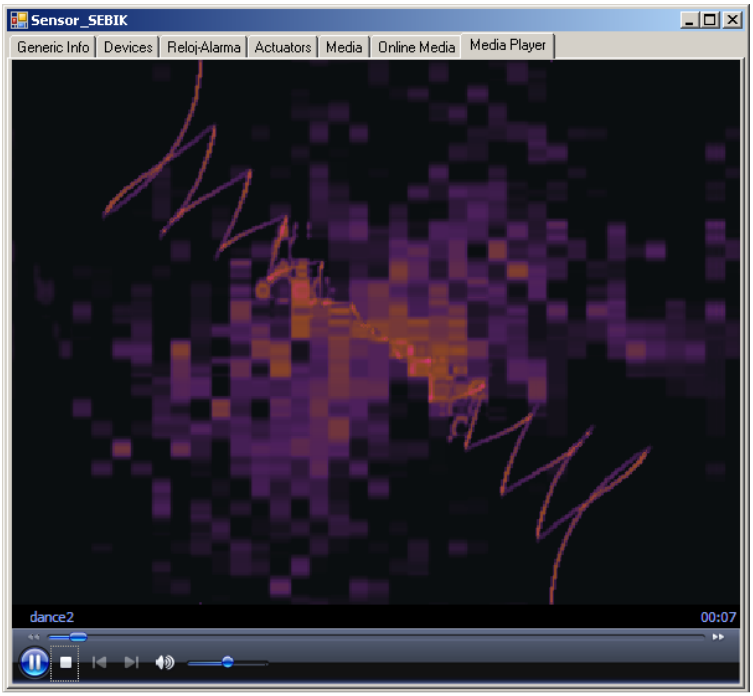


Fig. A.17 Vista del *Windows Media Player* en el editor

Servicio motor de reglas

Con el servicio ‘RulesEngine’ o motor de reglas se pretende simular el funcionamiento del motor de reglas en uno o más de un dispositivo (Fig. A.18), de tal forma, que el motor de reglas no esté ejecutándose en un único dispositivo, sino que pueda estar ejecutándose en varios dispositivos a la vez. La idea subyacente es que varios dispositivos sean capaces de controlar el entorno (coreografía), en vez de un único punto de control (orquestación). El modelo de orquestación describe un control único del comportamiento, mientras que la coreografía define un control distribuido. A través

de este servicio, se permitirá la creación de reglas, con tareas concretas que produzcan la invocación de acciones de diferentes dispositivos de la red.

Dependiendo de cada ubicación o instalación a realizar, a veces puede ser útil utilizar una arquitectura coreografiada en cuanto a distribución de reglas se refiere, al contrario de una arquitectura orquestada que puede ser la solución más común. Por ejemplo, en el caso de una red dinámica, donde los dispositivos aparecen y desaparecen continuamente (pierden la conexión), una arquitectura coreografiada puede resultar más eficaz para dar respuesta a las necesidades de los usuarios sin conectividad continua. La robustez de las comunicaciones en una arquitectura orquestada es un requisito necesario frente a las posibles desconexiones y zonas de dispositivos aisladas temporalmente que pudiera haber en una arquitectura coreografiada.

Para dar cabida a arquitecturas coreografiadas, se ha incluido dentro del editor el servicio Motor de Reglas. Tal y como se muestra en la figura A.18, se puede seleccionar el fichero de reglas deseado, cargarlo y finalmente activarlo. En cuanto a los dispositivos de la red, también se necesita descubrir los que se encuentran en el ámbito de actuación, por lo que incorpora un punto de control personalizado que localiza dispositivos UPnP de tipo ‘SEBIK’ únicamente, utilizando el botón ‘Load UPnP’ habilitamos esta función de punto de control. A través del botón ‘Load Devices’, en cambio, se da inicio a la búsqueda de dispositivos.

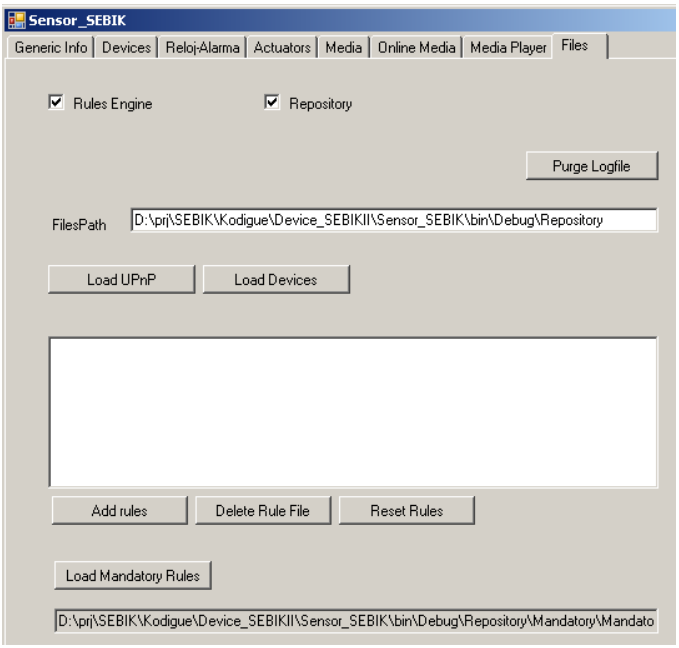


Fig. A.18 Vista de las variables del servicio *RulesEngine* en un dispositivo virtual

Queda en manos del instalador la elección de la configuración del despliegue de dispositivos y sus configuraciones. Dependerá de los recursos y capacidades de cada dispositivo y para garantizar las necesidades de los usuarios independientemente de la tipología de red que haya.

Finalmente, señalar también que, al ejecutarse los dispositivos virtuales en un PC, no existen limitaciones de recursos para esta primera aproximación, sea una arquitectura orquestada o una arquitectura coreografiada.

A.3 Dispositivo simulador para PDA

Siguiendo la misma estructura y especificación UPnP, se ha generado también el dispositivo con las categorías de dispositivos y servicios para PDA. El servicio de motor de reglas no se ha embebido en esta versión, pero es factible convertir la versión actual que funciona sobre el *Framework* 2.0 de .NET en PC a la plataforma Windows Mobile 6 para PDA en el futuro. La plataforma para la que se ha desarrollado el dispositivo para PDA ha sido Windows Mobile 6 [WM6]. Las pruebas se han realizado en una PDA Toshiba Portégé G900 [G900] conectandola vía WiFi con el resto de los dispositivos, editor de reglas y motor de reglas.

La diferencia significativa con la versión de PC es que la interfaz es básica, es decir, en esta versión no se ha generado una interfaz muy compleja, sino que se ha generado una interfaz UPnP a través de la que se puede interactuar con el dispositivo de la PDA como si se tratara de un dispositivo de PC.

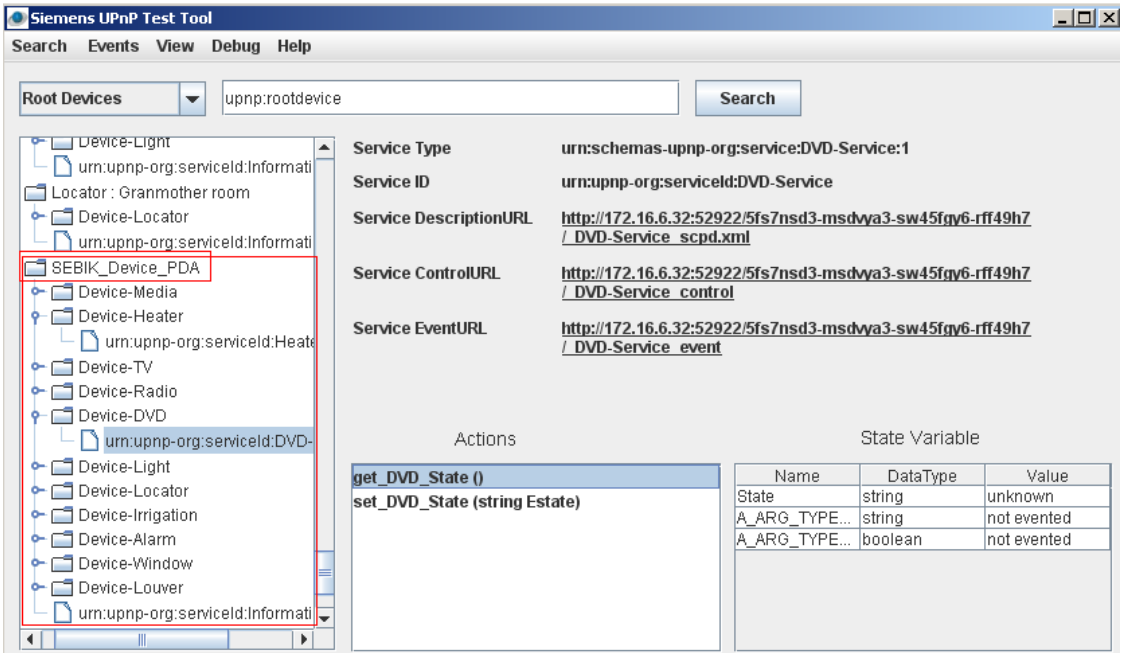


Fig. A.19 Vista de los servicios y variables del dispositivo PDA en el punto de control genérico de Siemens.

La utilización de este dispositivo sobre PDA va a ser de utilidad para validar cómo se comporta el MR ante la desconexión de un dispositivo en la red WiFi tal y cómo se estudia en el capítulo de validación de la tesis.

Anexo B: Motor de reglas

Este anexo pretende describir el motor de reglas. Un motor de reglas es un componente software concebido para satisfacer las necesidades del paradigma ECA *(Event-Condition-Action).* En nuestro caso, el motor de reglas será el encargado de interpretar las reglas de personalización del hogar y de ejecutarlas.

B.1 Introducción

Desde el Editor de Reglas (ER) o TAMAmI, los usuarios pueden crear sus reglas para personalizar su hogar. La creación de una regla consiste en definir su condición (o condiciones) y su acción (o acciones). El Motor de Reglas (MR) se encargará de procesar la información relativa a las reglas ECA definidas. Si en un momento dado se cumple alguna condición de las reglas definidas, el MR ejecuta la acción o las acciones dadas por las reglas afectadas. Para realizar esta tarea, el MR es capaz de descubrir dinámicamente todos los dispositivos disponibles y sus servicios a su alcance, según los servicios descubiertos y a la evaluación de las reglas de los usuarios, ejecuta las acciones correspondientes.

B.2 La interfaz

La interfaz del componente del MR está compuesta de varios elementos. Por una parte, se muestran los dispositivos descubiertos (Fig. B.1 derecha), por otra, se muestra la carga de ficheros de reglas (izquierda). Además, existe una serie de listas para mostrar los diferentes logs de los eventos analizados y de las acciones realizadas por el motor. El resto de comandos se explicarán a lo largo de este anexo. En la figura B.1 se muestra una vista general del motor de reglas.

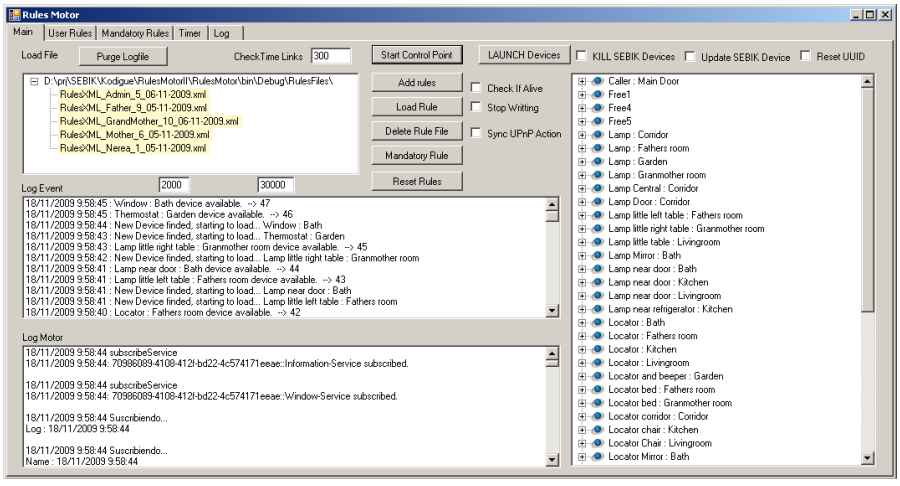


Fig. B.1 Vista principal del motor de reglas

B.2.1 Lista dispositivo

El MR mantiene en todo momento una lista de los dispositivos a su alcance. Cada vez que un dispositivo se anuncia o desaparece, la lista será actualizada. Pulsando sobre el botón ‘Start Control Point’ (Fig. B.1) el MR se pone a la escucha de mensajes de

anuncio de dispositivos en la red. En la figura B.2 se muestra un ejemplo de una lista de dispositivos encontrados en un momento cualquiera en la red.

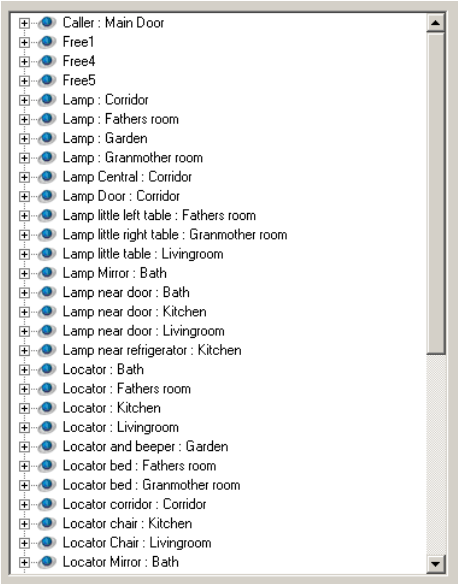


Fig. B.2 Vista parcial del motor de reglas de los dispositivos descubiertos

B.2.2 Listas de logs

El MRs contiene varias listas de logs. Estas listas de logs son imprescindibles para conocer la actuación y eficiencia del MR. En caso de errores, estos se visualizan en pantalla o se puede estar continuamente visualizando lo que está ocurriendo dentro del sistema (Fig. B.3).

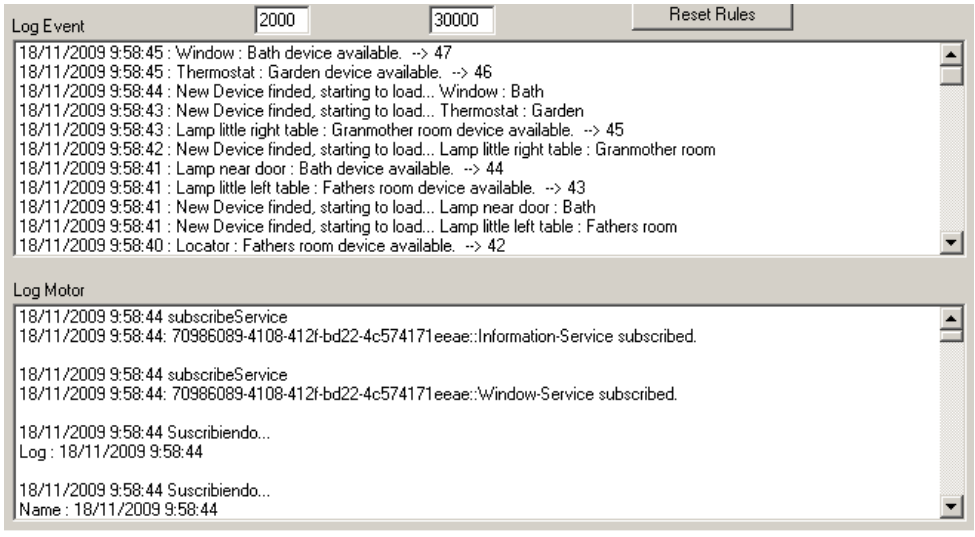


Fig. B.3 Vista parcial del motor de reglas del listado de los *logs*

Los logs también se guardan en ficheros para su posterior análisis más pormenorizado y la presentación de los resultados. Los valores de 2000 y 30000 que aparecen en la parte superior de la figura B.3 son para el mantenimiento de listas dentro de unos

rangos de valores óptimos, es decir, se pretende que las listas se mantengan en un número de eventos suficiente sin que crezcan indefinidamente y colapsen el sistema (Fig. B.3). En este caso, se mantienen en las listas los últimos 2000 valores. Y en cuanto al valor 30000 se trata de un segundo valor utilizado para el correcto funcionamiento del sistema. Si en algún momento en memoria se llega a ese número de eventos, directamente se guardan en los ficheros de los logs, sin visualizarlos, ya que la visualización cuando el sistema está sobrecargado ralentiza su actualización. En cualquier caso, estos valores son parámetros utilizados en desarrollo y depuración, y surgidos de la necesidad de respuesta del MR por debajo del segundo. El manejo de eventos por parte del MR se ha llevado a cabo a través de colas. Se han generado varias colas para los *logs* y procesos independientes para el manejo de las mismas. Por ejemplo, existe un proceso que se encarga del refresco de la pantalla y guardado en el disco duro de los *logs*. Otro proceso se encarga de la gestión de las reglas y de añadir los eventos ocurridos en las colas correspondientes. De esta manera, la tarea o proceso de visualización no interfiere con la tarea o proceso que se encarga de procesar los eventos recibidos de los dispositivos.

Existe la posibilidad de borrar todos los logs existentes en memoria y en pantalla y fichero, pulsando el botón de purga 'Purge LogFile' que se puede localizar en la parte superior izquierda de la figura B.1. Esta opción resulta útil cuando comenzamos con un nuevo test y no queremos mezclar datos de varios test en el tiempo.

B.2.3 Gestión de los ficheros de reglas

En la figura B.4.se muestra la interfaz del MR para la gestión de los ficheros de reglas. En ella se puede ver que haciendo clic en los botones correspondientes, podemos añadir ficheros de otras ubicaciones al directorio de trabajo, para a continuación cargarlos en el motor.

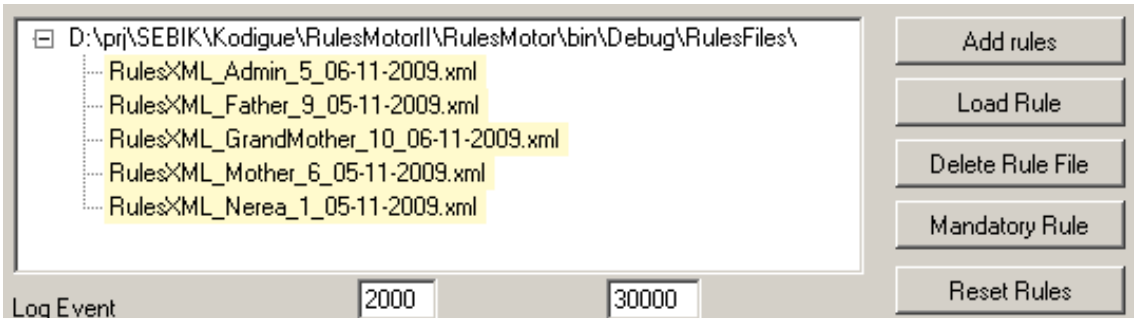


Fig. B.4 Vista parcial del motor de reglas de la carga de los ficheros de reglas

En el caso de que se quieran cargar otros ficheros de reglas, pulsando sobre el botón de 'Reset Rules', se eliminarán las reglas actuales cargadas por el MR, pero se mantendrá la lista de dispositivos.

B.2.4 Otras opciones

A continuación vamos a explicar el funcionamiento de otra serie de funcionalidades que tiene el MR. Estas funcionalidades vienen dadas en la figura B.5 recuadradas en rojo.

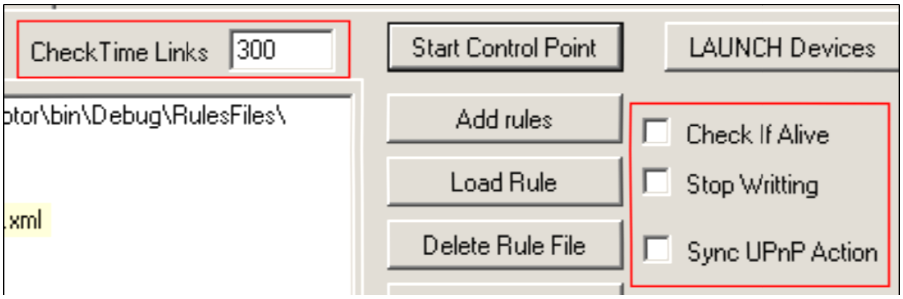


Fig. B.5 Vista parcial del motor de reglas de varias opciones

Chequeo de la consistencia

El MR incorpora dos mecanismos para chequear la conectividad y consistencia con los dispositivos enganchados y con las variables suscritas. Con la activación de la opción 'Check If Alive' mediante un mecanismo de 'ping' periódicamente se verifica que un dispositivo sigue vivo y al alcance. La forma de proceder es la siguiente, según el tiempo definido 'Check Time Links', en este caso cada 300 segundos, se realiza un barrido a todos los dispositivos que en teoría están online y registrados en la lista del MR. El barrido consiste en una llamada síncrona a la función de ping que incorporan todos los dispositivos. Si la llamada síncrona falla es porque no está al alcance, entonces para asegurarse de que todo va a funcionar como debe la próxima vez que aparezca, se elimina toda referencia al dispositivo. Esto es, se produce una de-suscripción a los eventos de las variables de este dispositivo que vienen dadas en las reglas definidas. Tal y cómo funcionan los mecanismos internos de suscripción utilizados en .NET, basta con que una llamada de ping falle para tener que rehacer todo mecanismo de suscripción anterior. No se puede garantizar que un dispositivo que se haya desconectado vuelva a conectarse utilizando los mismos 'puertos' o que siga utilizando los mismos. Es por ello que en el caso de que no hubiera respuesta al ping, el MR borraría al dispositivo correspondiente de su lista de dispositivos activos. Todos los elementos con asociaciones y suscripciones que tengan que ver con el dispositivo ausente tales como referencias al propio dispositivo y variables del motor de reglas se de-suscriben y se elimina toda información que hace referencia al mismo. Si el dispositivo volviera a conectarse a la red apareciendo como disponible, el MR volvería a realizar todo el proceso de suscripción como si fuera la primera vez.

Por otro lado, el mecanismo de 'Check If Alive' sirve para garantizar que toda variable involucrada en una condición siempre tiene un valor. Durante el desarrollo de esta tesis, se ha constatado que a veces las variables dejan de enviar eventos, independientemente de que el ping al dispositivo sea correcto. Es por ello que se ha implementado este mecanismo de ping a nivel de variable para conocer el estado del

mismo. Periódicamente, cada variable lógica verificará si su correspondiente variable real está viva (activa), o no (pasiva). Si se detecta que una variable deja de enviar cambios de los valores, pasará a un estado pasivo y, dependiendo de lo que el usuario haya especificado en el momento de la creación de la regla, la variable pasará a tomar un valor por defecto o seguirá manteniendo el último valor conocido. Es decir, las variables involucradas en una condición de un dispositivo que no está al alcance siempre van a tener un valor al menos por defecto. A la hora de crear las condiciones, el usuario podrá elegir entre utilizar un valor por defecto para los casos en los que el dispositivo no se encuentre disponible o que se utilice el último valor conocido. De esta manera, se garantiza que una regla compuesta por varias condiciones cuyas variables son dependientes de varios dispositivos, no falle en caso de que alguno de los dispositivos y/o variables no esté disponible. Habrá situaciones en que lo conveniente sea por seguridad pasar y utilizar cierto valor por defecto, y en otras seguir con el último valor bueno conocido.

Los mecanismos de 'ping' y 'Check If Alive' son complementarios. Con uno de los mecanismos conocemos si el dispositivo está ausente o no, y con el otro si las variables de un dispositivo están respondiendo bien o mal.

Pausar escritura en disco

La opción 'Stop Writting' sirve para no escribir los logs en el disco duro. Es decir, a veces puede interesar parar temporalmente la escritura en disco cara a acelerar las evaluaciones. Esta opción es útil sobre todo cuando realizamos pruebas de rendimiento del MR.

Acciones síncronas

La opción 'Sync UPnP Action' sirve para indicar al MR que las acciones que vaya a realizar sobre dispositivos UPnP las efectúe utilizando comunicaciones síncronas o asíncronas. Dependiendo del caso, puede interesar realizar unas llamadas síncronas (por ejemplo, si la red no es muy robusta) o asíncronas. En una comunicación asíncrona no obtendremos respuesta del resultado de cómo ha ido la actuación (por ejemplo, si se ha ejecutado la acción de la regla correctamente), mientras que en la comunicación síncrona conoceremos cómo ha ido y se podrá actuar en consecuencia. Por otro lado, si se han de ejecutar muchas acciones en poco tiempo o prácticamente simultáneamente, tal y como se pretende hacer con el ''action feeder'' (véase anexo C) para realizar las pruebas de validación como por ejemplo el 'test 2' del capítulo de validación, para generar acciones que ocurren en el mismo momento, la comunicación asíncrona parece la más adecuada, ya que la comunicación síncrona no se sabe de antemano cuánto tiempo va a consumir para ejecutar una secuencia de acciones predeterminada. La ejecución de acciones asíncronas en una red fiable y estable nos proporciona la posibilidad de ejecutar acciones prácticamente en tiempo real, según los datos obtenidos en las diferentes mediciones que se han realizado en la validación de la plataforma *middleware*.

Gestión de dispositivos

Cara a facilitar el manejo de dispositivos, que en un momento dado puede ser un número importante, se han creado varias utilidades que facilitan la carga y descarga de dispositivos y su actualización (Fig. B.6). Al igual que otra serie de características del MR, estas utilidades también cumplen una función facilitadora para la realización de los test de evaluación.

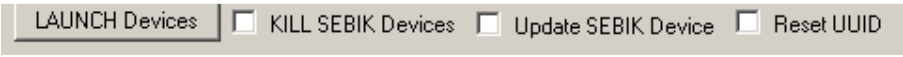


Fig. B.6 Vista parcial del motor de reglas, con las opciones para la gestión de los dispositivos

B.2.5 Vista de las reglas de los usuarios

Una vez cargadas las reglas de los usuarios, tal y como se ha explicado en el apartado 9.2.2.3, al seleccionar la pestaña 'User Rules' del menú principal del MR, se mostrarán las reglas cargadas en el editor (Fig. B.7). En la parte izquierda de esta figura se listan todas las reglas cargadas, y si se selecciona una de ellas, situándose en la raíz de la misma, en la parte superior derecha de la pantalla se mostrarán las acciones asociadas a dicha regla. Por otro lado, si expandimos la regla (haciendo clic sobre el signo +), también podremos ver cuál es la condición o condiciones que la componen. Finalmente, en la parte inferior derecha, existe una visualización de la parte de logs relativa a las reglas. En ella se muestra la información relativa a la recepción de eventos, afectación del evento a una regla, si la regla se ha satisfecho o no, si se han ejecutado las acciones, etc. La información de esta ventana y del log correspondiente será la que realmente utilizaremos para evaluar el MR.

Debido al gran número de eventos que pueden ocurrir, se utilizará Microsoft Excel para procesar la información de la evaluación. En el capítulo de validación se puede obtener más detalle sobre estos procedimientos. No obstante, en el propio MR y para facilitar la comprensión del funcionamiento del MR se ha creado un mecanismo para filtrar los datos. En la ventana 'Log' del MR se mostrará dicha información (Fig. B.8). Cada vez que pinchamos sobre una de las líneas de los logs de las reglas de usuario (en la parte inferior derecha de la figura B.7), se crea un filtro personalizado para todos los logs que hayan ocurrido con el mismo número de hilo de ejecución.

Entre la información que se loga o registra se encuentran la hora de ocurrencia del evento, el número de hilo en el que se ha ejecutado, donde normalmente cada evento de entrada se ejecuta en un hilo diferente (asignado automáticamente por el sistema operativo), con lo que todas las acciones desencadenadas dentro del motor de reglas debidas a un único evento las podemos agrupar identificando su número de hilo. Es por ello que utilizaremos este número de hilo para poder agrupar eventos desencadenados por el mismo evento de entrada y ver la secuencia que siguen todos los eventos en la secuencia de ocurrencia. El resto de campos utilizados en el log se analizarán a posteriori en este documento.

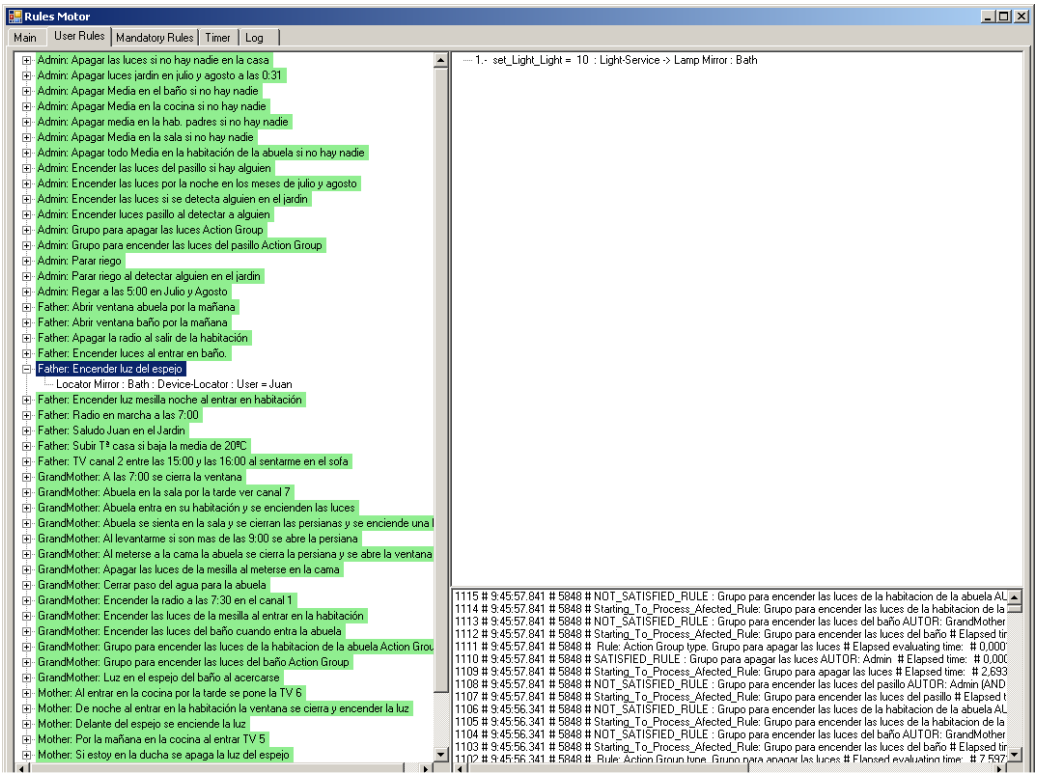


Fig. B.7 Vista de las reglas de los usuarios

En la figura B.8 se muestra como ejemplo los logs ocurridos para la ejecución del hilo número 5468.

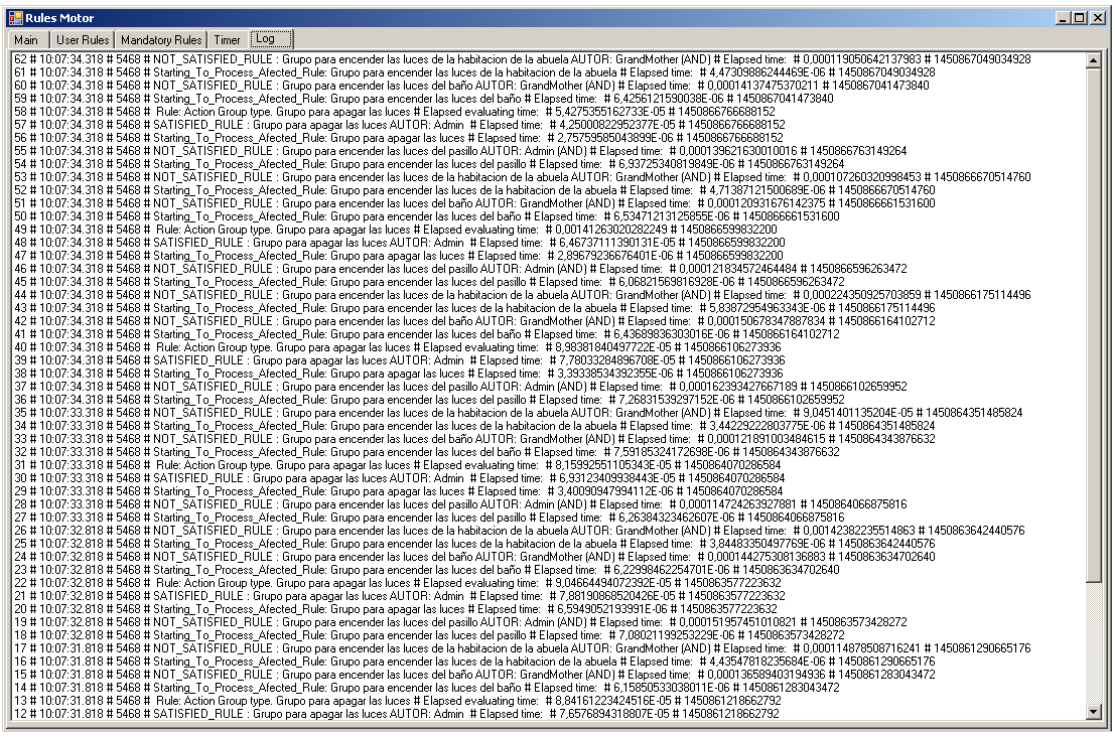


Fig. B.8 Vista *logs* filtrados

B.2.6 Vista de las reglas del sistema o mandatory

Para las reglas de tipo ‘mandatory’ o del sistema, se dispone también de un visualizador de reglas particular. Al igual que lo visto para las reglas de usuario, las reglas del sistema se visualizan junto con las acciones y eventos relacionados con este tipo de reglas. En la figura B.9 se puede observar la lista de reglas cargadas, y qué aspecto tiene para una regla en particular.

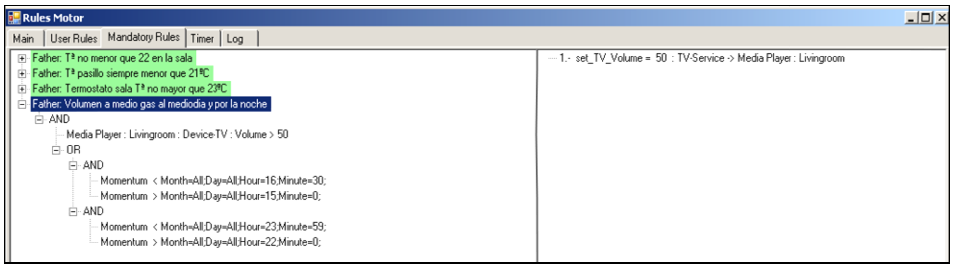


Fig. B.9 Vista de las reglas del sistema o ‘mandatory’

Desde el punto de vista de las reglas y en cuanto a su visualización, tal y como se observa, la visualización de las reglas de los usuarios y del sistema es idéntica. La diferencia está en cómo trata el motor de reglas dichas reglas en el momento de la ejecución. Más adelante en este anexo se explica el funcionamiento y como se evalúan un tipo de reglas frente a las otras.

B.2.7 Reloj interno

El motor de reglas dispone de una utilidad para poder personalizar el reloj o el tiempo que se usa como referencia. Aunque normalmente este reloj será igual al del PC o dispositivo en el que se ejecuta, puede que por ejemplo, para realizar simulaciones sea interesante utilizar otra referencia de tiempo. La realización de simulaciones en un período de tiempo menor al real puede ser interesante. En este sentido, por ejemplo, el poder simular las acciones que ocurrirían en 24 horas, en solamente 24 minutos es interesante desde el punto de vista del tiempo que se ahorra en cada simulación.

En la figura B.10 se muestran las opciones de configuración del reloj, donde si optamos por personalizar el reloj, podremos elegir la nueva hora de referencia y el tipo de frecuencia, normal o acelerado. Únicamente se podrán seleccionar estos dos tipos de frecuencias. El modo acelerado sirve para realizar simulaciones. Se trata de acelerar el tiempo para simular las acciones que ocurrirían en 24 horas en 24 minutos.

En la figura B.10 se puede observar que en la parte inferior derecha (en la barra de tareas del sistema) se encuentra enmarcada en rojo la hora real del PC donde se ejecuta el MR, y en el centro de pantalla la hora que utiliza como referencia el MR.

La utilidad de esta característica se analiza y se explica en detalle en el capítulo de la validación en su capítulo de la tesis.

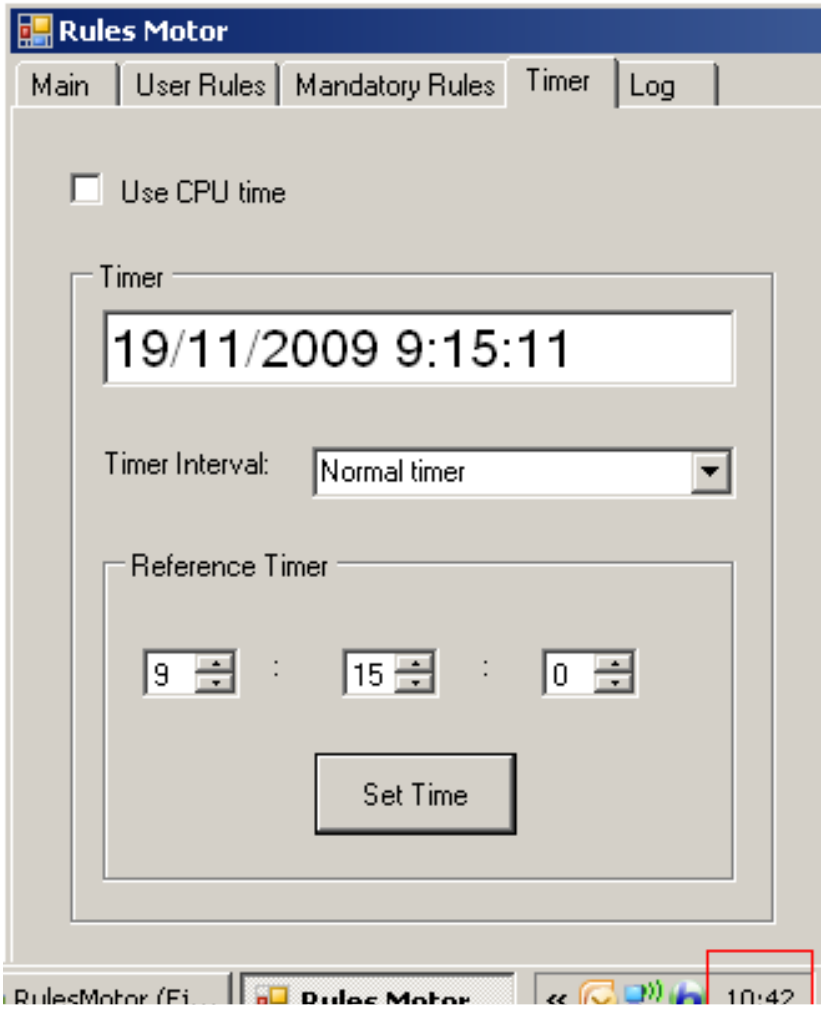


Fig. B.10 Vista de las opciones para la configuración del reloj o tiempo de referencia

B.3 Funcionamiento del MR

Una vez que se han cargado las reglas en el MR y descubierto los dispositivos de la red, el MR comienza la evaluación de los eventos que llegan. A continuación analizaremos con varios ejemplos el funcionamiento interno del MR.

Dentro del MR, los elementos de tipo variables, condiciones y reglas se asocian de tal forma que entre ellos se intercambia información por medio de eventos. Así, si una variable de un dispositivo está incluida en una condición y su valor ha cambiado, un evento es recibido por la condición a la que pertenece cada vez que ese cambio ocurre. Lo mismo ocurre entre las condiciones y las reglas. Vamos a analizar dicho paso de eventos entre las componentes de una regla utilizando varias condiciones. Véase ‘capítulo 5 Configurabilidad’ para más detalle.

La condición de tipo simple u ordinaria consta del UUID del dispositivo, el servicio afectado, la variable y el valor a evaluar. Por otra parte, la condición de tipo fecha consta de los parámetros de tiempo que se deseen evaluar (esto es, mes, día, hora y minuto).

Un ejemplo de una regla con condición simple sería ‘cuando Garbiñe esté en el salón’, si el sensor de presencia del salón tiene UUID ‘a2484ae7-51ce-498b-aa53-00f28baf936e’, la condición quedaría expresada en el ER y en el MR de la manera siguiente:

```
"(a2484ae7-51ce-498b-aa53-00f28baf936e::Servicio-Locator::User = Garbiñe)"
```

En el caso de una condición de tipo tiempo, ésta sería:

```
Date <= Month=All;Day=All;Hour=7;Minute=0;)
```

Un ejemplo de una regla con una condición compuesta es

```
"(AND     (Date <= Month=All;Day=All;Hour=7;Minute=0;)
(Date >= Month=All;Day=All;Hour=1;Minute=0;)
(a2484ae7-51ce-498b-aa53-00f28baf936e::Servicio-Locator::User = Garbiñe)
)"
"(AND     (cf909169-11c7-4902-b3b8-ed5bdbb8c475::Servicio-Locator::User = Garbiñe)
(AND
(Date >= Month=August;Day=All;Hour=15;Minute=0;)
(Date <= Month=August;Day=All;Hour=18;Minute=0;)
)
)"
```

Según se va creando la estructura interna de las reglas, las variables involucradas en las condiciones de las reglas se van encadenando para que produzcan y comuniquen los cambios de estado a quien corresponda. Por cada variable real de los dispositivos en el motor de reglas se crea su variable gemela. A esta variable la denominaremos variable lógica. La variable real y lógica estarán enlazadas de manera que cuando ocurra un cambio en la variable del dispositivo, en su variable lógica del motor de reglas también se recibirá dicho cambio.

Si por ejemplo se recibe un evento de cambio en una variable que forma parte de una regla, los eventos generados serán los siguientes:

1. Evento que llega a la variable lógica del MR desde el dispositivo real.
2. Evento que llega a la condición correspondiente desde la variable lógica.
3. Evento que llega a la regla correspondiente desde una de sus condiciones.
4. Se evalúa la regla: se evalúan todas las condiciones.

5. La evaluación de las condiciones se lleva a cabo para todas las condiciones de una regla, independientemente de si el valor de una variable de una condición ha sido la causante de la activación de la evaluación o no.
6. Si la parte de la condición se satisface, el ejecutor de acciones recibirá un evento para que se ejecuten las acciones asociadas a la regla que se acaba de satisfacer.
7. Se realizan las comprobaciones de prioridades y permisos pertinentes si la regla afectada es de un usuario y se ejecutan las acciones.
8. Se comprueba si hay conflicto con alguna otra regla cargada en el MR de otro usuario. Se evalúan entre el set de reglas de usuarios cargados, aquellas que también contengan la acción que se pretende llevar a cabo. En caso afirmativo, el valor de la acción será el definido por la regla que tenga la mayor prioridad.
9. Se comprueba si hay conflicto con alguna regla del sistema. Al igual que en el caso anterior, se evalúan entre el set de reglas del sistema aquellas en las que se encuentre la acción que queremos modificar, si se cumple alguna regla, el valor de la actuación será aquella que marque la regla del sistema satisfecha. En caso negativo, el valor inicial no se verá afectado.

Los conflictos se resuelven comprobando la prioridad. La acción que tenga la mayor prioridad es la que se va a ejecutar en caso de conflicto.

B.4 Casos prácticos

B.4.1 Ejemplo 1

"Cuando Garbiñe entre en la sala, que se encienda solamente la luz de la mesilla al 50% de su potencia y que se encienda la televisión en el canal 5 al 60% de volumen."

En la figura B.11 se muestra la regla creada en el ER:

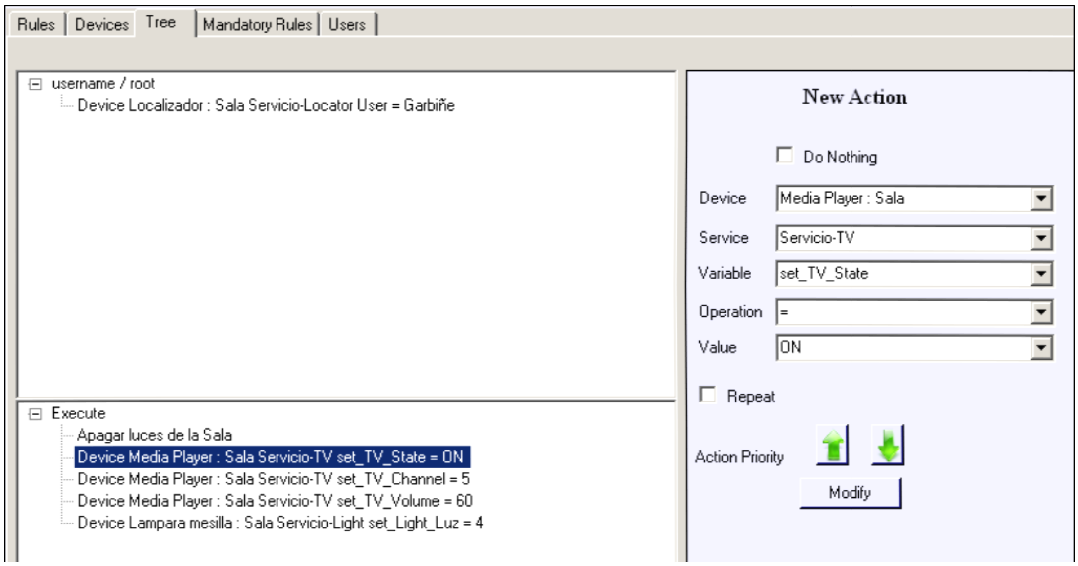


Fig. B.11 Vista de la regla en el editor de reglas

Y la misma regla cargada en el MR se muestra en la figura B.12:

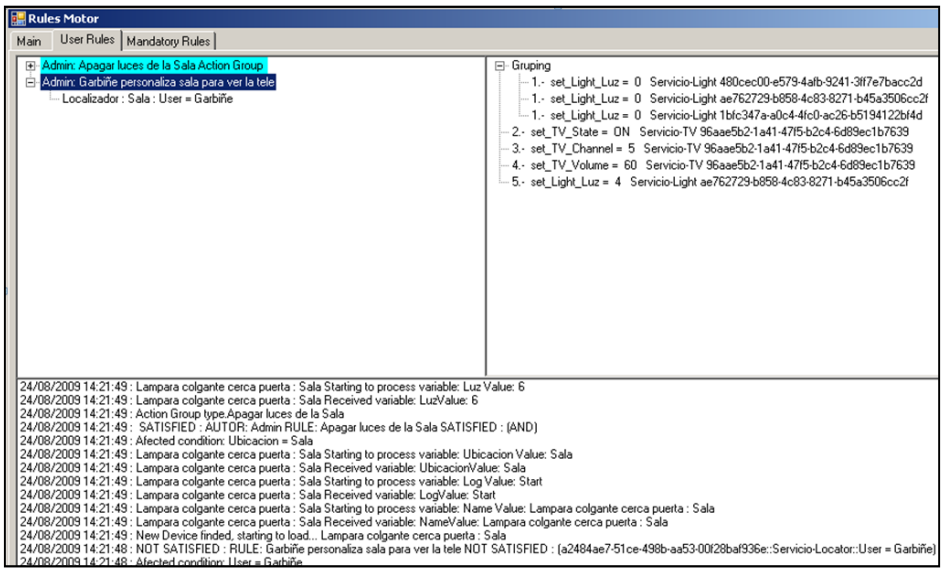


Fig. B.12 Vista de la regla en el motor de reglas

En la parte superior izquierda de la figura B.12 se muestra la parte condicional de las reglas cargadas. En la parte superior derecha la lista de acciones a realizar si se cumple con la condición. También, se puede apreciar en la parte superior derecha que actualmente existen tres dispositivos detectados que ofrecen luz debido a que entre las acciones existe una que es de tipo grupo.

A continuación en la figura B.13 vamos a ver cómo reacciona el motor si simulamos que “Andrés” entra en la sala:

Fig. B.13 Vista del log del MR al producirse el evento de que Andrés entra en la sala

En los *logs* de la figura anterior, se puede ver que se ha recibido el cambio de la variable del localizador de la sala y que en principio afecta a una de las reglas, pero cuya condición no se cumple.

A continuación, simulamos que entra el usuario Garbiñe en la sala y, como consecuencia, se producen los eventos que han sido recogidos en el log del MR que se muestra en la figura B.14.

En este caso, ante un cambio del usuario, el sistema reacciona de manera diferente, tal y como muestran los logs de la figura anterior. La regla se ha satisfecho y, en consecuencia, ejecuta las acciones. Destacar que, por ejemplo, la televisión ya se encontraba en el canal 5 por lo que no se ha realizado esa acción, evitando tráfico y reduciendo el tiempo de respuesta, es decir, antes de actuar sobre una variable, el motor comprueba el valor real, únicamente lo actualiza si es diferentes. Por otro lado, se puede ver que la ejecución de las acciones produce reacciones, es decir, los

dispositivos mandan a su vez que ha existido un cambio de variable, con lo que el motor trata de evaluar dicho cambio. En este caso, al no estar asociados con ninguna regla no producen ningún efecto.

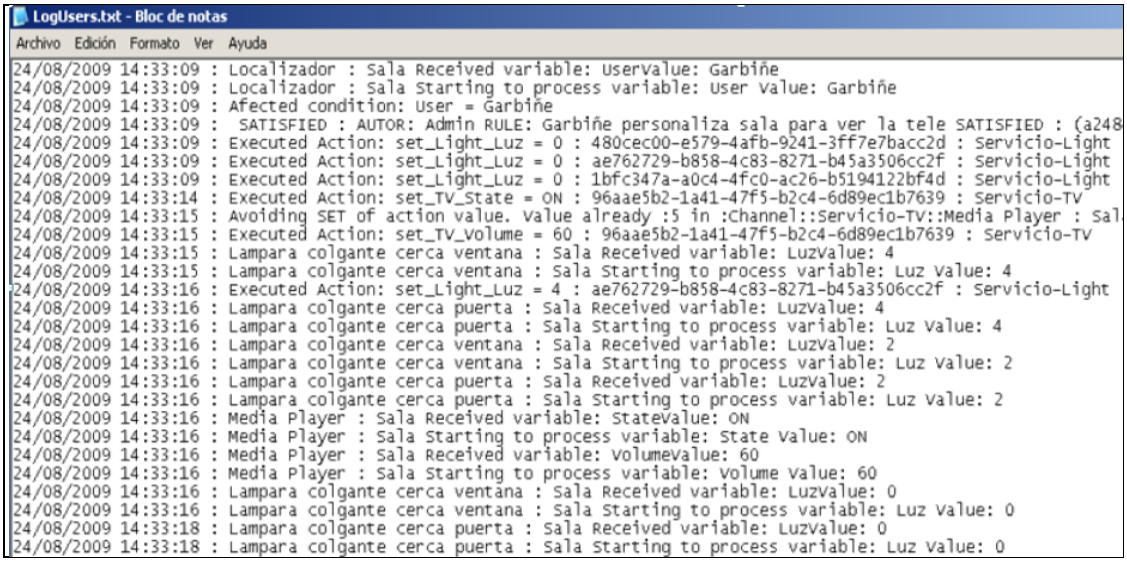


Fig. B.14 Vista de log de eventos recibidos

Si volvemos a simular que el usuario Garbiñe entra de nuevo en la sala, se producen los log que se muestran en la figura B.15.

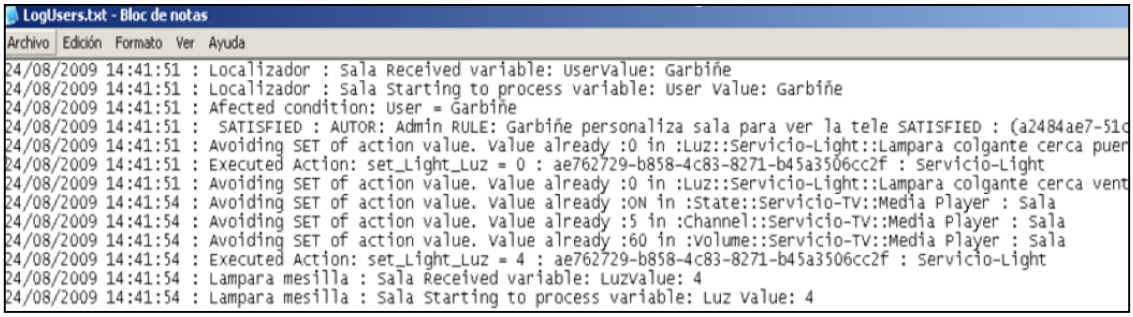


Fig. B.15 Vista de log de eventos recibidos

En este caso, en la lista de logs producidos mostrados en la figura anterior se aprecia que: exclusivamente apaga una de las lámparas, la única que estaba encendida y que, a la hora de actualizar los valores deseados como son los mismos, se evita la actualización de los mismos. Únicamente se enciende la lámpara seleccionada. En este caso, para garantizar que la única lámpara que se queda encendida es la de la mesilla, primero de forma global se apagan todas las lámparas de la sala y seguido se enciende la que nos interesa.

B.4.2 Ejemplo 2

Para provocar un conflicto entre la regla anterior, vamos a cargar ahora en el MR una regla del sistema para que entre las dos de la tarde y las cuatro de la tarde no se pueda poner el volumen de la televisión más alto que el 30% (figura B.16).

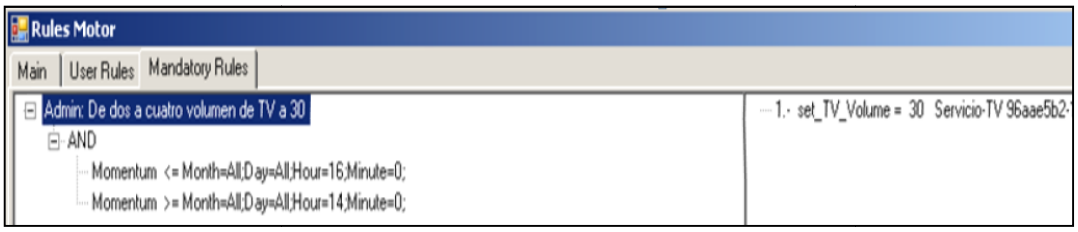


Fig. B.16 Vista de regla del sistema en el editor de reglas

En consecuencia, si ahora volvemos a hacer que Garbiñe entre en la sala, se producen los eventos que se muestran en la siguiente figura B.17.

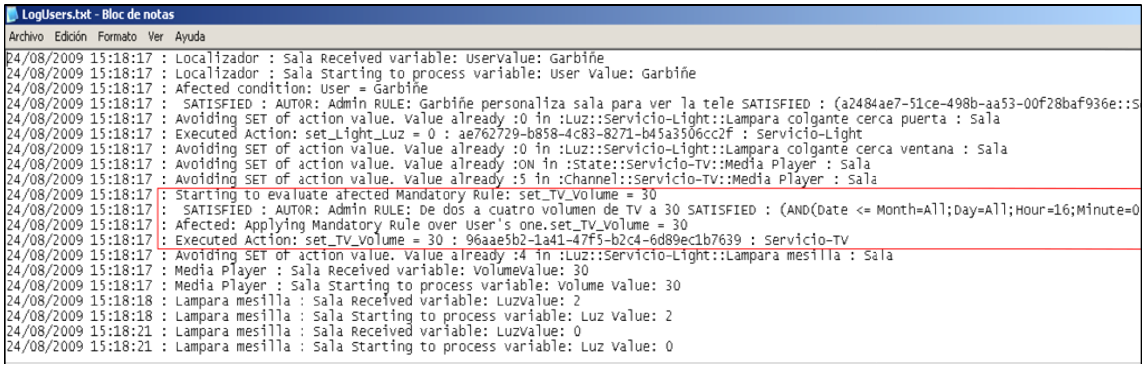


Fig. B.17 Vista de log de eventos recibidos

En este caso, tal y como se muestra en rojo en la figura B.17, el MR detecta que hay una regla del sistema afectada cuando el usuario intenta poner el volumen de la televisión al valor deseado. Como se observa en la imagen anterior, se evalúa la regla del sistema y, como es satisfactoria, el valor que se utiliza como parámetro para asignar al volumen es el que indica la regla del sistema en este caso.

B.4.3 Ejemplo 3

Vamos a crear una regla para el usuario padre que quiere que entre las 15:00 y las 15:45 siempre el canal de televisión sea la primera cadena para poder ver las noticias. La lista de reglas cargadas en el MR quedaría como se muestra en la figura B.18:

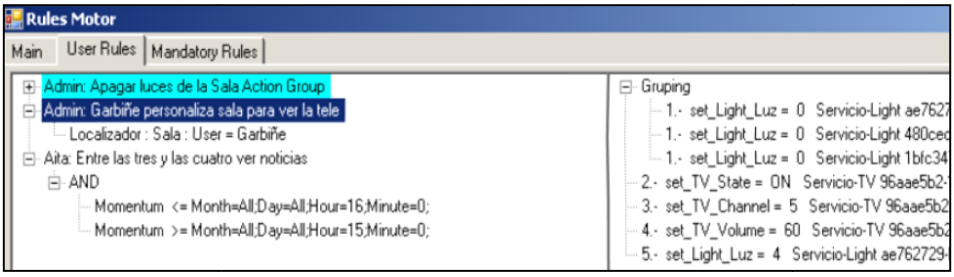


Fig. B.18 Vista de regla en el editor de reglas

Hasta el momento se han cargado reglas de dos usuarios, con prioridades diferentes y las reglas del sistema. Al asignarle mayor prioridad a una de ellas, el motor a la hora

de aplicar cambios, debería hacer caso a la regla creada por el padre. Vamos a verificarlo. Si simulamos que Garbiñe entra en la sala a las 15:22 horas, el canal de televisión que debería verse es la primera cadena por ser la que tiene establecido el padre en su regla con mayor prioridad.

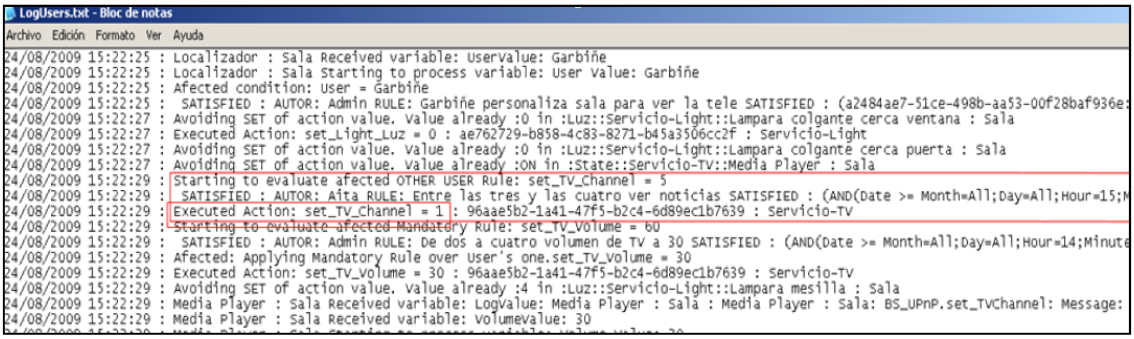


Fig. B.19 Vista de log de los eventos recibidos.

Tal y como se ha señalado, en la lista de logs producidos que se muestran en la figura B.19, el MR detecta un conflicto entre reglas y aplica el valor de aquella regla que tiene mayor prioridad.

Anexo C: Action Feeder

En este anexo se describe el 'action feeder' que permitirá la ejecución controlada de acciones para poder realizar los test de validación.

C.1 Action Feeder

El Editor de Reglas (ER) incorpora un generador de acciones que permitirá la validación del motor de reglas (MR). Para que una regla se valide, es necesario que se produzcan eventos o, lo que es lo mismo, cambios de variables de los dispositivos. Gracias al generador, se va a poder definir una lista de acciones a invocar que provocarán dichos eventos. Esta invocación de acciones se corresponde a la secuencia de acciones que un usuario o los miembros de una familia podrían llevar a cabo en su casa o bien que producen otros cambios de variables de los dispositivos existentes. Por una parte, el *'action feeder'* permitirá la definición de una lista de acciones (apartado 1.1), por otra parte dichas acciones se podrán agrupar (apartado 1.2) y, finalmente, se podrá definir la secuencia de las acciones indicando el tiempo o intervalo de tiempos que han de transcurrir entre las mismas (apartado 1.3).

C.1.1 Creación de acciones

Para crear una acción simulada, de la lista de dispositivos disponibles, se seleccionará la acción deseada y se asignará el valor de la variable que se quiera utilizar en la simulación.

En la figura C.1, a la izquierda, tenemos el listado de todos los dispositivos disponibles. Una vez seleccionado el dispositivo y desplegando la información que ofrece, seleccionaremos la acción que se desea realizar. Tras seleccionar la variable a través de la su función asociada que comienza con 'set_', en la parte derecha de la figura C.1, se podrá introducir el valor que se desee.

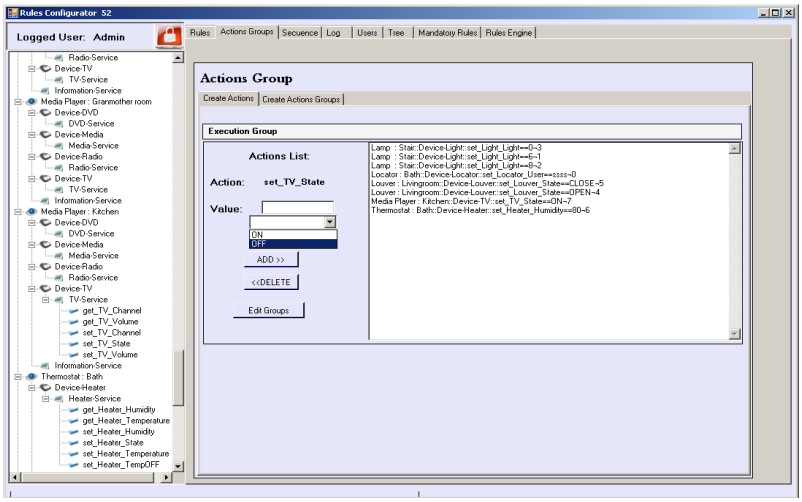


Fig. C.1 Vista de la lista de acciones definidas

C.1.2 Agrupación de acciones

La agrupación de acciones es útil para poder realizar ejecuciones de acciones en el mismo instante de tiempo. Por ejemplo, la inicialización de todas las variables de todos los dispositivos para cada ciclo de evaluación se realiza por una agrupación de acciones. También es útil para realizar una acción común (apagar o encender) sobre un mismo tipo de dispositivos, por ejemplo, para apagar todas las lámparas. En este último caso se agruparan todos los dispositivos de tipo luz con el valor de apagado.

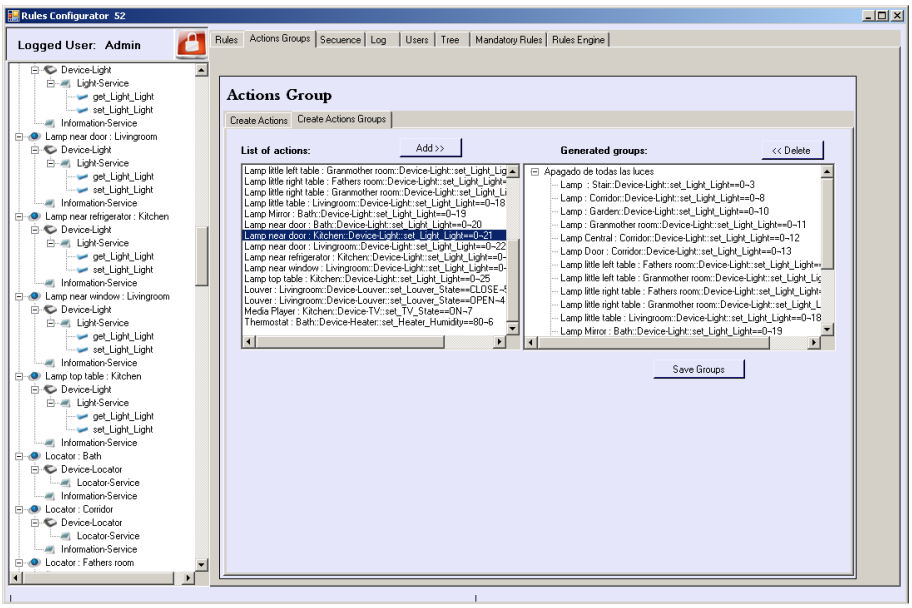


Fig. C.2 Vista de la lista de acciones y los grupos de acciones definidos

En la figura C.2, en la parte central se encuentra la lista de todas las acciones creadas, y en la parte derecha de la figura se encuentran los grupos de acciones que se han ido creando. La formación de grupos se realiza primero creando el grupo utilizando el botón derecho del ratón. Seguidamente se le asigna el nombre del grupo y, finalmente, utilizando los botones de añadir y eliminar, seleccionando en uno y otro lado de las dos listas de acciones y grupos, se van añadiendo o quitando acciones de los grupos.

C.1.3 Secuenciación y ejecución de acciones

Una vez creadas las acciones y los grupos de acciones, se ha de especificar la secuencia en que los grupos de acciones se tienen que ejecutar durante la simulación. En la figura C.3 se muestra que partiendo de los grupos de acciones que se han generado anteriormente (en la parte izquierda), se añaden las acciones en el orden que se quieran ejecutar. Posteriormente, también se puede modificar el orden de un grupo subiéndolas y bajándolas en la lista utilizando las flechas correspondientes. Una vez ordenados los grupos de acciones según cómo se desee que sucedan en el tiempo, se deberá añadir en qué momento se desea que ocurran dichos grupos de acciones. Para ello, si se selecciona cada elemento de la lista de simulación, se podrá añadir o modificar la hora de ejecución en el recuadro 'Set Time'.

Respecto al valor de la hora de ejecución, se supuso que se fuera a realizar a lo largo de un día (24 horas). Por simplificación, la notación de la hora va de 0 a 23 y los minutos de 0 a 99. Por lo tanto, si se pretende que una acción se ejecute a la una y cuarto del mediodía, en el simulador se deberá indicar como tiempo 13,25, y si se desea que se ejecute a la una y media del mediodía, se indicará 13,5.

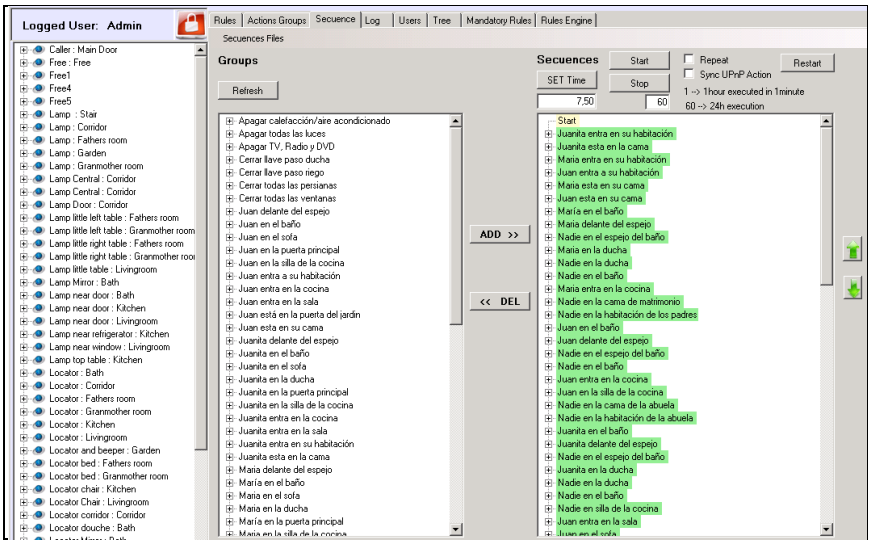


Fig. C.3 Vista de una secuencia de acciones a ejecutar por el *'action feeder'*

Por otro lado, para poder validar el MR se dispone de la posibilidad de realizar la ejecución de las acciones en vez de en horas en minutos (Fig. C.3). Por ejemplo, las acciones de 1 día entero (24 horas) se podrían ejecutar en 24 minutos. Para ello basta con añadir el número de la escala que se quiere utilizar en la casilla correspondiente.

Esto puede ser interesante para hacer un seguimiento más rápido de la simulación de las acciones. El botón ‘Start’ dará comienzo a la simulación. Si en cualquier momento se desea detener/pausar la ejecución, se pulsará sobre el botón de ‘Stop’.

También, se dispone de un modo de repetición para ejecutar una y otra vez toda la secuencia de acciones y así estudiar el comportamiento del MR en periodos largos de tiempo, como si se simularan varios días consecutivos.

Por último, señalar que se puede configurar el ‘action feeder’ para que la ejecución de las acciones realizadas se lleven a cabo utilizando mecanismos de comunicación síncronos o asíncronos.

Anexo D: Cuestionario de validación

Código: .....

D.1 Explicación

Con este cuestionario se pretende estudiar la utilidad del editor de reglas y su manejo por usuarios con diferentes perfiles. Los resultados obtenidos ayudarán a valorar su potencial uso en el hogar, su facilidad de manejo y las opiniones de los mismos también servirán para analizar sus mejoras.

El cuestionario consta de tres partes. La primera relativa a los **datos personales** a rellenar por el usuario. Después se le enseñará cómo usar el editor de reglas. En la segunda parte se realizarán las **tareas encomendadas** (creación de 4 reglas) y, finalmente, en la tercera parte, el usuario rellenará una pequeña encuesta con sus **impresiones**.

D.2 Datos personales

El propósito de este breve formulario es recoger información básica acerca de usted. Esto nos ayudará a comparar a los participantes en todas las pruebas que estamos llevando a cabo y ver si existen diferencias entre ellos. **Toda la información que proporcione será estrictamente confidencial.**

1. ¿En qué año nació?

Nací en el año.....

2. ¿Cuál es su género/sexo?

Masculino □

Femenino □

3. ¿Cuál es o ha sido su ocupación laboral?

.....

4. ¿Cuál es su nivel educativo acabado más alto?

Estudios primarios □

Estudios secundarios □

Estudios universitarios □

Doctorado □

Otros (por favor especifique) □

.....

5. ¿Con qué frecuencia utiliza los ordenadores?

Nunca □

Una vez a la semana □

Varias veces a la semana □

Todos los días □

¿En el trabajo o en casa o en ambos sitios? .....

Algún comentario: .....

6. ¿Sabe programar?

Sí □

No □

Si la respuesta es sí ¿qué lenguajes de programación conoce? .....

7. ¿Cuántos años lleva programando?

Número de años .....

8. ¿Te consideras un usuario con conocimientos técnicos en informática?

Sí □

No □

9. ¿Sueles realizar instalaciones de programas en el ordenador?

Sí □

No □

D.3 Uso del editor de reglas

A continuación, el instructor te explicará el propósito del editor de reglas y la creación de dos reglas (una regla con una única condición y acción y otra regla con dos condiciones y dos acciones).

D.4 Tareas a realizar

A continuación dispóngase a generar las siguientes reglas:

Hora inicio: .....

Regla 1

Cuando el usuario ‘Juan’ entre en la cocina, que se enciendan las luces y se abra la persiana.

Regla 2

Si la temperatura media de la sala, cocina y pasillo es menor que 18ºC, poner la calefacción en marcha a una temperatura de 22ºC en la sala y cocina, y a 20ºC en el pasillo.

Regla 3

Si el usuario ‘Nerea’ entra en la sala entre las 17:00 y 20:00, que se ponga la televisión en marcha en el canal 5, con el volumen al 40%. Que se atenúen todas las luces de la sala también.

Regla 4

Cuando no haya ‘nadie’ en casa que se apaguen todas la luces.

Hora Fin: .....

D.5 Cuestionario postest

A continuación, se te realizarán unas preguntas generales sobre el sistema y después otras relacionadas con la creación de las reglas y el diseño del propio editor (interfaz).

D.5.1 Preguntas generales

1. ¿Crees que sería interesante personalizar el funcionamiento de tu hogar? Que el hogar se comporte como tú quieres

Sí □

No □

Más o menos □

Comentarios:.....

2. ¿Crees que el editor de reglas es útil para realizarlo?

Sí □

No □

Más o menos □

Comentarios:.....

3. ¿Te ha parecido fácil el manejo del editor de reglas?

Muy fácil □

Fácil □

Ni fácil ni difícil □

Difícil □

Muy difícil □

Comentarios:.....

4. ¿Te ha parecido sencillo y comprensible su manejo? ¿Por qué?

Comentarios:.....

5. ¿Qué añadirías o quitarías del editor de reglas?

.....

6. ¿Qué regla o reglas te han parecido más interesantes para que existieran en tu hogar?

.....

7. ¿Alguna otra regla que sería muy interesante para ti?

.....

8. ¿Alguna mejora sobre el uso o manejo del editor?

.....

D.5.2 Preguntas sobre el diseño del editor

PARTE 1: Información sobre la creación de condiciones

1. ¿Te ha parecido fácil crear las condiciones?

Muy fácil .....

Fácil.....

Ni fácil ni difícil.....

Difícil.....

Muy difícil .....

Comentarios: .....

2. ¿Te ha parecido intuitiva la creación de las condiciones? ¿Cambiarías algo?

.....

3. ¿Se te ocurre una condición que no se contempla en el editor de reglas? ¿Cuál?

.....

4. ¿Qué quitarías o añadirías para mejorar el proceso de creación de condiciones?

.....

PARTE 2: Información sobre la creación de las acciones

1. ¿Te ha parecido fácil crear las acciones?

Muy fácil .....

Fácil.....

Ni fácil ni difícil.....

Difícil.....

Muy difícil .....

Comentarios: .....

2. ¿Te ha parecido intuitiva la creación de las acciones? ¿Cambiarías algo?

.....

3. ¿Se te ocurre una acción que no se contempla en el editor de reglas? ¿Cuál?

.....

4. ¿Qué quitarías o añadirías para mejorar el proceso de creación de acciones?

.....

PARTE 3: COMENTARIOS EN GENERAL

1. ¿Algún comentario?

.....

.....

.....

D.5.3 Datos a rellenar por el encuestador

1. ¿Ha sido capaz el usuario de crear todas las reglas?

Sí □

No □ ¿Cuáles? …….

2. ¿Ha sido capaz el usuario de crear todas las condiciones?

Sí □

Sí, con cierta ayuda □

Sí, con mucha ayuda □

No □

3. ¿Ha sido capaz el usuario de crear todas las acciones?

Sí □

Sí, con cierta ayuda □

Sí, con mucha ayuda □

No □

Anexo E: Resultados test “Un día en casa”

En este anexo se describen los resultados obtenidos como consecuencia de haber realizado la simulación de acciones y movimientos que los miembros de una casa hubieran realizado a lo largo de un día cualquiera. También se muestran las acciones resultantes que el motor de reglas ha producido como respuesta a las necesidades planteadas por los usuarios en sus reglas.

E.1 Introducción

A continuación se mostrarán los diferentes filtros que se han aplicado utilizando MS Excel. A la hora de generar la información de log, se han utilizado palabras clave o identificativas de la acción o evento que se pretende registrar. Creando filtros utilizando estas palabras clave, se obtendrán los datos resumen.

Los datos que en este documento se muestran son la consecuencia de haber simulado 79 acciones para 66 reglas, que se han ejecutado durante la interpretación de los movimientos de los usuarios de la casa para el periodo de un día.

En las figuras que se utilizan a continuación, las columnas muestran la siguiente información:

a. Columna A: los datos de la primera columna indican el número de log. Se trata de un número que el motor de reglas asigna secuencialmente.
b. Columna B: la segunda columna recoge la hora o momento de la ejecución respecto al reloj de simulación o utilizado en el motor de reglas como referencia. Véase anexo A para más información respecto a este reloj.
c. Columna C: la tercera columna muestra la hora real del dispositivo en el que se está ejecutando el motor de reglas.
d. Columna D: la cuarta columna muestra la descripción del log.
e. Columna E: la quinta columna muestra el tiempo transcurrido desde la llegada del cambio de variable que comenzó la ejecución de la tarea hasta el momento que ocurrió el log.

f. Columna F: la sexta columna muestra el tiempo de referencia del ciclo del reloj de la CPU tomada en el momento de la llegada del cambio de variable.

Debido a la gran cantidad de *logs*, y a los diferentes filtros generados, para poder diferenciarlos visualmente más fácilmente, se han coloreado los diferentes tipos de filtros utilizados. Tal y como se muestra en la figura E.1, a cada tipo de log se le ha asignado un color. En dicha figura se muestran todos los *logs* según han ocurrido en el tiempo. Por ejemplo, comentar que para el test de un día, se han contabilizado 1057 *logs*. Dependiendo del número de reglas, número de dispositivos y las acciones simuladas, dicho valor variará. De todas formas, para realizar cualquier tipo de análisis o valoración sobre un fichero de *logs* que consta de muchas líneas, una línea tras otra, la utilización de herramientas de filtrado se hace imprescindible.

	A	B	C	D	E	F
202	200	7:55:31.0	10:44:28.240	Starting_To_Process_Afected_Rule: Encender luz del espejo	0,0000443693	25317911836179
203	201	7:55:31.0	10:44:28.240	NOT_SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-	0,0001503439	25317911836179
204	202	7:55:31.0	10:44:28.240	Starting_To_Process_Afected_Rule: Luz en el espejo del baÃ±o al acercarse	0,0001667226	25317911836179
205	203	7:55:31.0	10:44:28.240	NOT_SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8	0,0002348600	25317911836179
206	204	7:55:31.0	10:44:28.240	Starting_To_Process_Afected_Rule: Delante del espejo se enciende la luz	0,0002483365	25317911836179
207	205	7:55:31.0	10:44:28.240	NOT_SATISFIED_RULE : Delante del espejo se enciende la luz AUTOR: Mother (4b8961a8-	0,0003166514	25317911836179
208	206	7:59:31.0	10:44:32.93	Received_Variable: Light : Value: 2 from: Lamp Mirror : Bath	25329432792033,0000000000	25329432792033
209	207	7:59:31.0	10:44:32.93	Starting_To_Process variable: Light Value: 2 of device: Lamp Mirror : Bath	0,0000244328	25329432792033
210	208	8:00:31.0	Alguna de las de DATE de la regla estÃ¡ activa : 3e139e62-885e-4336-8202-41093028f635			
211	209	8:00:31.0	10:44:33.926	Starting_To_Process_Afected_Rule: Al levantarme si son mas de las 9:00 se abre la pers	0,0000261981	25334895758625
212	210	8:00:31.0	10:44:33.926	NOT_SATISFIED_RULE : Al levantarme si son mas de las 9:00 se abre la persiana AUTOR:	0,0003527442	25334895758625
213	211	8:01:31.0	10:44:34.505	Received_Variable: Light : Value: 2 from: Lamp little left table : Fathers room	25336649591586,0000000000	25336649591586
214	212	8:01:31.0	10:44:34.505	Starting_To_Process variable: Light Value: 2 of device: Lamp little left table : Fathers roc	0,0000258222	25336649591586
215	213	8:06:31.0	10:44:39.1	Received_Variable: User : Value: None from: Locator : Bath	25350056622117,0000000000	25350056622117
216	214	8:06:31.0	10:44:39.1	Starting_To_Process variable: User Value: None of device: Locator : Bath	0,0000237952	25350056622117
217	215	8:06:31.0	10:44:39.1	Starting_To_Process_Afected_Rule: Apagar Media en el baÃ±o si no hay nadie	0,0000401649	25350056622117
218	216	8:06:31.0	10:44:39.1	SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (975abbc7-b0	0,0004529263	25350056622117
219	217	8:06:31.0	Execute Action List: ccfbe7db-185d-4613-8900-360dbfe72034 --> 25350056622117			
220	218	8:06:31.0	10:44:39.1	Starting to set action: set_TV_State = OFF : Device-TV : TV-Service	0,0006138113	25350056622117
221	219	8:06:31.0	10:44:39.1	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Device-TV	0,0006809713	25350056622117
222	220	8:06:31.0	10:44:39.1	Starting to set action: set_Radio_State = OFF : Device-Radio : Radio-Service	0,0007134431	25350056622117
223	221	8:06:31.0	10:44:39.1	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio	0,0007579959	25350056622117
224	222	8:06:31.0	10:44:39.1	Starting to set action: set_DVD_State = OFF : Device-DVD : DVD-Service	0,0007860617	25350056622117
225	223	8:06:31.0	10:44:39.1	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Device-DVD	0,0008299679	25350056622117
226	224	8:06:31.0	End process....			
227	225	8:06:31.0	10:44:39.1	Starting_To_Process_Afected_Rule: Apagar las luces del baÃ±o cuando no haya nadie	0,0008579254	25350056622117
228	226	8:06:31.0	10:44:39.1	SATISFIED_RULE : Apagar las luces del baÃ±o cuando no haya nadie AUTOR: Admin (975a	0,0011566221	25350056622117
229	227	8:06:31.0	Execute Action List: d5fa5913-772a-4062-9e90-3ae589613832 --> 25350056622117			
230	228	8:06:31.0	10:44:39.1	Starting to set action: set_Light_Light = 0 : Device-Light : Light-Service	0,0015692692	25350056622117
231	229	8:06:31.0	10:44:39.32	Action executed = set_Light_Light : Device-Light : Light-Service	0,0316571528	25350056622117
232	230	8:06:31.0	10:44:39.32	Starting to set action: set_Light_Light = 0 : Device-Light : Light-Service	0,0320287448	25350056622117
233	231	8:06:31.0	10:44:39.32	Action executed = set_Light_Light : Device-Light : Light-Service	0,0331383270	25350056622117
234	232	8:06:31.0	End process....			
235	233	8:06:31.0	10:44:39.32	Starting_To_Process_Afected_Rule: Encender luces al entrar en baÃ±o.	0,0331766723	25350056622117
236	234	8:06:31.0	10:44:39.32	NOT_SATISFIED_RULE : Encender luces al entrar en baÃ±o. AUTOR: Father (975abbc7-b0b8	0,0335259489	25350056622117
237	235	8:06:31.0	10:44:39.32	Starting_To_Process_Afected_Rule: Abrir ventana baÃ±o por la maÃ±ana	0,0335408419	25350056622117
238	236	8:06:31.0	10:44:39.32	NOT_SATISFIED_RULE : Abrir ventana baÃ±o por la maÃ±ana AUTOR: Father (AND(975abb	0,0338231479	25350056622117

Fig. E.1 Vista general del *log*

De todos los log que se muestran en la figura E.1, a continuación se han creado vistas específicas para los más significativos.

E.2 Cambios de variable

En la siguiente figura E.2 se muestran como ejemplo una muestra de variables recibidas.

	A	B	C	D	E	F
22	182	7:47:31.0	10:44:20.611	Received_Variable: User : Value: Juan from: Locator Mirror : Bath	25295147012340	25295147012340
23	194	7:52:31.0	10:44:25.263	Received_Variable: Light : Value: 2 from: Lamp near door : Bath	25309036841058	25309036841058
24	196	7:53:31.0	10:44:26.78	Received_Variable: Light : Value: 4 from: Lamp little left table : Fathers room	25311468186630	25311468186630
25	198	7:55:31.0	10:44:28.240	Received_Variable: User : Value: None from: Locator Mirror : Bath	25317911836179	25317911836179
26	206	7:59:31.0	10:44:32.93	Received_Variable: Light : Value: 2 from: Lamp Mirror : Bath	25329432792033	25329432792033
27	211	8:01:31.0	10:44:34.505	Received_Variable: Light : Value: 2 from: Lamp little left table : Fathers room	25336649591586	25336649591586
28	213	8:06:31.0	10:44:39.1	Received_Variable: User : Value: None from: Locator : Bath	25350056622117	25350056622117
29	241	8:06:31.0	10:44:39.847	Received_Variable: Light : Value: 4 from: Lamp near door : Bath	25352598139002	25352598139002
30	243	8:08:31.0	10:44:41.491	Received_Variable: Light : Value: 4 from: Lamp Mirror : Bath	25357495420296	25357495420296
31	245	8:10:31.0	10:44:43.841	Received_Variable: Light : Value: 6 from: Lamp near door : Bath	25364514238380	25364514238380
32	247	8:13:31.0	10:44:46.754	Received_Variable: Light : Value: 0 from: Lamp little left table : Fathers room	25373212595145	25373212595145
33	249	8:17:31.0	10:44:50.91	Received_Variable: Light : Value: 6 from: Lamp Mirror : Bath	25383161380338	25383161380338
34	251	8:23:31.0	10:44:56.623	Received_Variable: User : Value: Juan from: Locator chair : Kitchen	25402649095053	25402649095053
35	253	8:23:31.0	10:44:56.952	Received_Variable: Light : Value: 8 from: Lamp near door : Bath	25403666613336	25403666613336
36	255	8:27:31.0	10:45:00.993	Received_Variable: Light : Value: 10 from: Lamp Mirror : Bath	25415715227598	25415715227598
37	257	8:28:31.0	10:45:01.933	Received_Variable: Light : Value: 10 from: Lamp near door : Bath	25418500358292	25418500358292
38	259	8:29:31.0	10:45:02.465	Received_Variable: User : Value: None from: Locator bed : Granmother room	25420084904799	25420084904799
39	271	8:29:31.0	10:45:02.857	Received_Variable: Light : Value: 8 from: Lamp Mirror : Bath	25421297015178	25421297015178
40	273	8:34:31.0	10:45:07.776	Received_Variable: Light : Value: 8 from: Lamp near door : Bath	25435971848331	25435971848331
41	275	8:44:31.0	10:45:17.832	Received_Variable: Light : Value: 6 from: Lamp near door : Bath	25465992750936	25465992750936
42	277	8:45:31.0	10:45:18.803	Received_Variable: Light : Value: 6 from: Lamp Mirror : Bath	25468897998429	25468897998429
43	279	8:46:31.0	10:45:19.179	Received_Variable: Light : Value: 4 from: Lamp Mirror : Bath	25470023291073	25470023291073
44	281	8:48:31.0	10:45:21.466	Received_Variable: User : Value: None from: Locator : Granmother room	25476808760613	25476808760613
45	309	8:49:31.0	10:45:22.907	Received_Variable: Light : Value: 4 from: Lamp near door : Bath	25481136288816	25481136288816
46	311	8:51:31.0	10:45:24.943	Received_Variable: Light : Value: 2 from: Lamp near door : Bath	25487232824151	25487232824151
47	313	8:57:31.0	10:45:31.5	Received_Variable: Light : Value: 2 from: Lamp Mirror : Bath	25505307108846	25505307108846
48	315	8:58:31.0	10:45:32.102	Received_Variable: User : Value: Juanita from: Locator : Bath	25508573378847	25508573378847
49	337	9:03:31.0	10:45:36.550	Received_Variable: Light : Value: 0 from: Lamp Mirror : Bath	25521837381180	25521837381180
50	339	9:07:31.0	10:45:40.858	Received_Variable: Light : Value: 0 from: Lamp near door : Bath	25534700577576	25534700577576
51	341	9:09:31.0	10:45:42.659	Received_Variable: User : Value: Juanita from: Locator Mirror : Bath	25540100090424	25540100090424
52	357	9:17:31.0	10:45:50.147	Received_Variable: User : Value: None from: Locator Mirror : Bath	25562458656378	25562458656378

Fig. E.2 Vista variables recibidas

E.3 Reglas afectadas

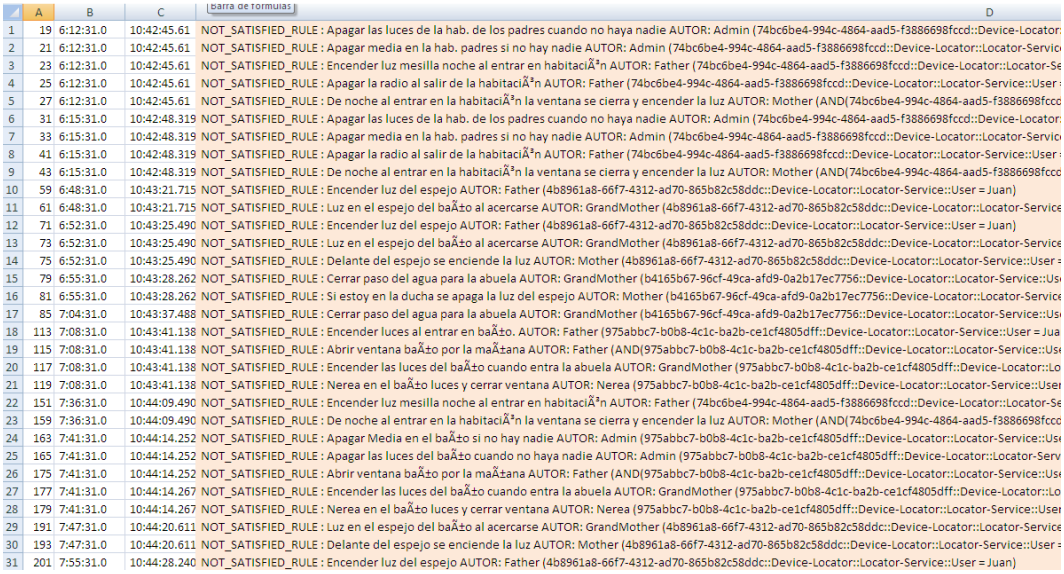
En la siguiente figura E.3 se muestran como ejemplo la lista de reglas que han sido afectadas por los cambios de variable recibidos.

	A	B	C	D	E	F
1	1	0:31:31.0	10:37:02.614	Starting_To_Process_Afected_Rule: Apagar luces jardin en julio y agosto a las 0:31	0,0000346732	23987671804251
2	8	5:00:31.0	10:41:33.398	Starting_To_Process_Afected_Rule: Regar a las 5:00 en Julio y Agosto	0,0000445768	24795978561954
3	18	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Apagar las luces de la hab. de los padres cuando no haya nadie	0,0000459272	25009896405177
4	20	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Apagar media en la hab. padres si no hay nadie	0,0002114769	25009896405177
5	22	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Encender luz mesilla noche al entrar en habitaciÃ³n	0,0003267204	25009896405177
6	24	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: Apagar la radio al salir de la habitaciÃ³n	0,0004408482	25009896405177
7	26	6:12:31.0	10:42:45.61	Starting_To_Process_Afected_Rule: De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz	0,0005561670	25009896405177
8	30	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Apagar las luces de la hab. de los padres cuando no haya nadie	0,0000411694	25019627919165
9	32	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Apagar media en la hab. padres si no hay nadie	0,0002005958	25019627919165
10	34	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Encender luz mesilla noche al entrar en habitaciÃ³n	0,0003097312	25019627919165
11	40	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: Apagar la radio al salir de la habitaciÃ³n	0,0072158344	25019627919165
12	42	6:15:31.0	10:42:48.319	Starting_To_Process_Afected_Rule: De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz	0,0073533453	25019627919165
13	58	6:48:31.0	10:43:21.715	Starting_To_Process_Afected_Rule: Encender luz del espejo	0,0000489978	25119309532311
14	60	6:48:31.0	10:43:21.715	Starting_To_Process_Afected_Rule: Luz en el espejo del baÃ±o al acercarse	0,0002174918	25119309532311
15	62	6:48:31.0	10:43:21.715	Starting_To_Process_Afected_Rule: Delante del espejo se enciende la luz	0,0002975690	25119309532311
16	70	6:52:31.0	10:43:25.490	Starting_To_Process_Afected_Rule: Encender luz del espejo	0,0000419994	25130577171609
17	72	6:52:31.0	10:43:25.490	Starting_To_Process_Afected_Rule: Luz en el espejo del baÃ±o al acercarse	0,0001583016	25130577171609
18	74	6:52:31.0	10:43:25.490	Starting_To_Process_Afected_Rule: Delante del espejo se enciende la luz	0,0002352751	25130577171609
19	78	6:55:31.0	10:43:28.262	Starting_To_Process_Afected_Rule: Cerrar paso del agua para la abuela	0,0000432686	25138874992527
20	80	6:55:31.0	10:43:28.262	Starting_To_Process_Afected_Rule: Si estoy en la ducha se apaga la luz del espejo	0,0000844801	25138874992527
21	84	7:04:31.0	10:43:37.488	Starting_To_Process_Afected_Rule: Cerrar paso del agua para la abuela	0,0000513497	25166427775569
22	86	7:04:31.0	10:43:37.488	Starting_To_Process_Afected_Rule: Si estoy en la ducha se apaga la luz del espejo	0,0000941972	25166427775569
23	94	7:08:31.0	10:43:41.138	Starting_To_Process_Afected_Rule: Apagar Media en el baÃ±o si no hay nadie	0,0000443182	25177308542046
24	104	7:08:31.0	10:43:41.138	Starting_To_Process_Afected_Rule: Apagar las luces del baÃ±o cuando no haya nadie	0,0008970196	25177308542046
25	112	7:08:31.0	10:43:41.138	Starting_To_Process_Afected_Rule: Encender luces al entrar en baÃ±o.	0,0034273677	25177308542046
26	114	7:08:31.0	10:43:41.138	Starting_To_Process_Afected_Rule: Abrir ventana baÃ±o por la maÃ±ana	0,0037687586	25177308542046
27	116	7:08:31.0	10:43:41.138	Starting_To_Process_Afected_Rule: Encender las luces del baÃ±o cuando entra la abuela	0,0040789590	25177308542046
28	118	7:08:31.0	10:43:41.138	Starting_To_Process_Afected_Rule: Nerea en el baÃ±o luces y cerrar ventana	0,0043788678	25177308542046
29	130	7:36:31.0	10:44:09.490	Starting_To_Process_Afected_Rule: Apagar las luces de la hab. de los padres cuando no haya nadie	0,0000497557	25261943802201
30	140	7:36:31.0	10:44:09.490	Starting_To_Process_Afected_Rule: Apagar media en la hab. padres si no hay nadie	0,0031347581	25261943802201
31	150	7:36:31.0	10:44:09.490	Starting_To_Process_Afected_Rule: Encender luz mesilla noche al entrar en habitaciÃ³n	0,0035231649	25261943802201

Fig. E.3 Vista reglas afectadas

E.4 Reglas no satisfechas

En la siguiente figura E.4 se muestran como ejemplo unas cuantas reglas no satisfechas.

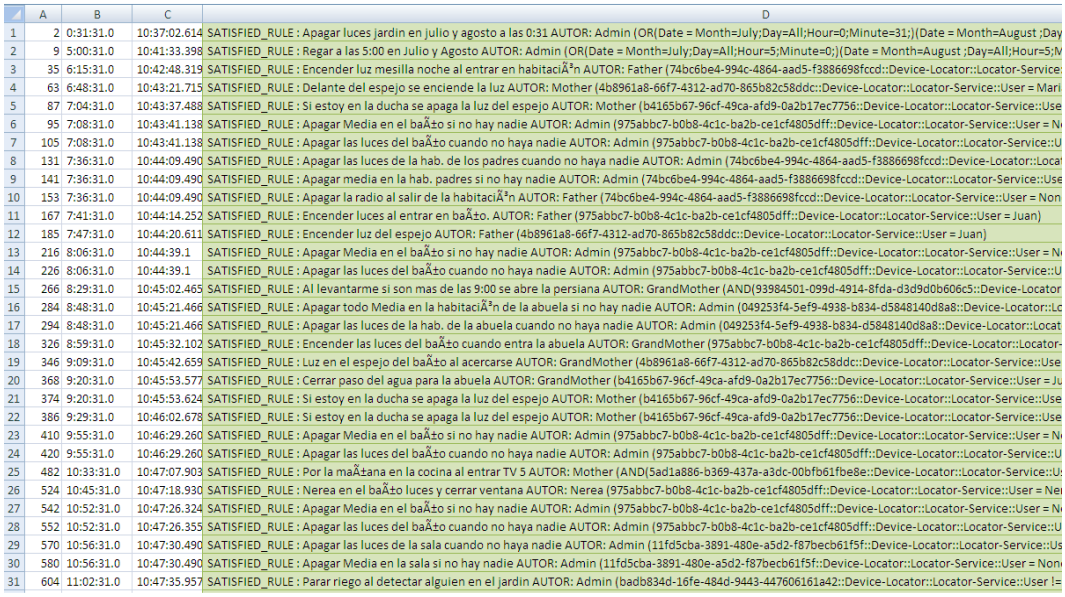


	A	B	C	D
1	19	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator:
2	21	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Servic
3	23	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Se
4	25	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : Apagar la radio al salir de la habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Service::User =
5	27	6:12:31.0	10:42:45.61	NOT_SATISFIED_RULE : De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz AUTOR: Mother (AND(74bc6be4-994c-4864-aad5-f3886698fccc
6	31	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator:
7	33	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Servic
8	41	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : Apagar la radio al salir de la habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Service::User =
9	43	6:15:31.0	10:42:48.319	NOT_SATISFIED_RULE : De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz AUTOR: Mother (AND(74bc6be4-994c-4864-aad5-f3886698fccc
10	59	6:48:31.0	10:43:21.715	NOT_SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Juan)
11	61	6:48:31.0	10:43:21.715	NOT_SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service
12	71	6:52:31.0	10:43:25.490	NOT_SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Juan)
13	73	6:52:31.0	10:43:25.490	NOT_SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service
14	75	6:52:31.0	10:43:25.490	NOT_SATISFIED_RULE : Delante del espejo se enciende la luz AUTOR: Mother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User =
15	79	6:55:31.0	10:43:28.262	NOT_SATISFIED_RULE : Cerrar paso del agua para la abuela AUTOR: GrandMother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::Use
16	81	6:55:31.0	10:43:28.262	NOT_SATISFIED_RULE : Si estoy en la ducha se apaga la luz del espejo AUTOR: Mother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service
17	85	7:04:31.0	10:43:37.488	NOT_SATISFIED_RULE : Cerrar paso del agua para la abuela AUTOR: GrandMother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::Use
18	113	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Encender luces al entrar en baÃ±o. AUTOR: Father (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = Jua
19	115	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Abrir ventana baÃ±o por la maÃ±ana AUTOR: Father (AND(975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::Use
20	117	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Encender las luces del baÃ±o cuando entra la abuela AUTOR: GrandMother (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Lo
21	119	7:08:31.0	10:43:41.138	NOT_SATISFIED_RULE : Nerea en el baÃ±o luces y cerrar ventana AUTOR: Nerea (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User
22	151	7:36:31.0	10:44:09.490	NOT_SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Se
23	159	7:36:31.0	10:44:09.490	NOT_SATISFIED_RULE : De noche al entrar en la habitaciÃ³n la ventana se cierra y encender la luz AUTOR: Mother (AND(74bc6be4-994c-4864-aad5-f3886698fccc
24	163	7:41:31.0	10:44:14.252	NOT_SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::Use
25	165	7:41:31.0	10:44:14.252	NOT_SATISFIED_RULE : Apagar las luces del baÃ±o cuando no haya nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Serv
26	175	7:41:31.0	10:44:14.252	NOT_SATISFIED_RULE : Abrir ventana baÃ±o por la maÃ±ana AUTOR: Father (AND(975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::Use
27	177	7:41:31.0	10:44:14.267	NOT_SATISFIED_RULE : Encender las luces del baÃ±o cuando entra la abuela AUTOR: GrandMother (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Lo
28	179	7:41:31.0	10:44:14.267	NOT_SATISFIED_RULE : Nerea en el baÃ±o luces y cerrar ventana AUTOR: Nerea (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User
29	191	7:47:31.0	10:44:20.611	NOT_SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service
30	193	7:47:31.0	10:44:20.611	NOT_SATISFIED_RULE : Delante del espejo se enciende la luz AUTOR: Mother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User =
31	201	7:55:31.0	10:44:28.240	NOT_SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Juan)

Fig. E.4 Vista reglas no satisfechas

E.5 Reglas satisfechas

En la siguiente figura E.5 se muestran como ejemplo unas cuantas reglas satisfechas.

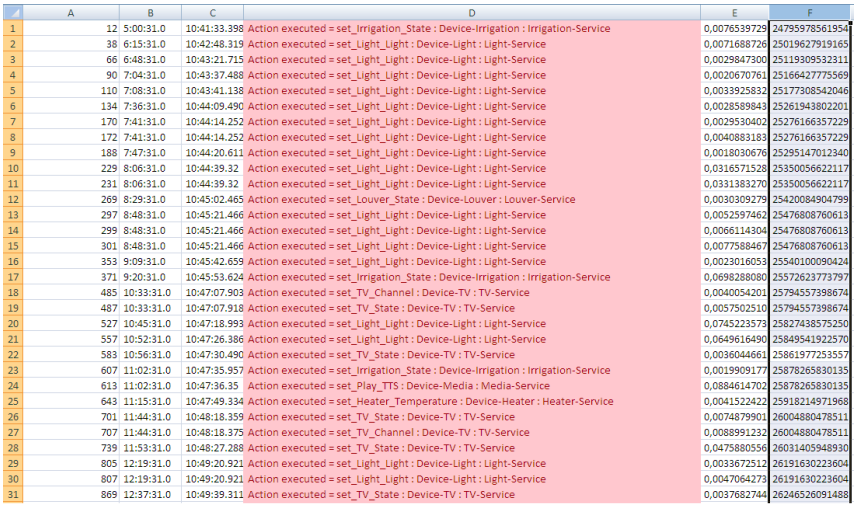


	A	B	C	D
1	2	0:31:31.0	10:37:02.614	SATISFIED_RULE : Apagar luces jardin en julio y agosto a las 0:31 AUTOR: Admin (OR(Date = Month=July;Day=All;Hour=0;Minute=31;)(Date = Month=August ;Day
2	9	5:00:31.0	10:41:33.398	SATISFIED_RULE : Regar a las 5:00 en Julio y Agosto AUTOR: Admin (OR(Date = Month=July;Day=All;Hour=5;Minute=0;)(Date = Month=August ;Day=All;Hour=5;M
3	35	6:15:31.0	10:42:48.319	SATISFIED_RULE : Encender luz mesilla noche al entrar en habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Service:
4	63	6:48:31.0	10:43:21.715	SATISFIED_RULE : Delante del espejo se enciende la luz AUTOR: Mother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Mari
5	87	7:04:31.0	10:43:37.488	SATISFIED_RULE : Si estoy en la ducha se apaga la luz del espejo AUTOR: Mother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::Use
6	95	7:08:31.0	10:43:41.138	SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = N
7	105	7:08:31.0	10:43:41.138	SATISFIED_RULE : Apagar las luces del baÃ±o cuando no haya nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::U
8	131	7:36:31.0	10:44:09.490	SATISFIED_RULE : Apagar las luces de la hab. de los padres cuando no haya nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locat
9	141	7:36:31.0	10:44:09.490	SATISFIED_RULE : Apagar media en la hab. padres si no hay nadie AUTOR: Admin (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Service::Use
10	153	7:36:31.0	10:44:09.490	SATISFIED_RULE : Apagar la radio al salir de la habitaciÃ³n AUTOR: Father (74bc6be4-994c-4864-aad5-f3886698fccd::Device-Locator::Locator-Service::User = Non
11	167	7:41:31.0	10:44:14.252	SATISFIED_RULE : Encender luces al entrar en baÃ±o. AUTOR: Father (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = Juan)
12	185	7:47:31.0	10:44:20.611	SATISFIED_RULE : Encender luz del espejo AUTOR: Father (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::User = Juan)
13	216	8:06:31.0	10:44:39.1	SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = N
14	226	8:06:31.0	10:44:39.1	SATISFIED_RULE : Apagar las luces del baÃ±o cuando no haya nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::U
15	266	8:29:31.0	10:45:02.465	SATISFIED_RULE : Al levantarme si son mas de las 9:00 se abre la persiana AUTOR: GrandMother (AND(93984501-099d-4914-8fda-d3d9d0b606c5::Device-Locator
16	284	8:48:31.0	10:45:21.466	SATISFIED_RULE : Apagar todo Media en la habitaciÃ³n de la abuela si no hay nadie AUTOR: Admin (049253f4-5ef9-4938-b834-d5848140d8a8::Device-Locator::Lo
17	294	8:48:31.0	10:45:21.466	SATISFIED_RULE : Apagar las luces de la hab. de la abuela cuando no haya nadie AUTOR: Admin (049253f4-5ef9-4938-b834-d5848140d8a8::Device-Locator::Locat
18	326	8:59:31.0	10:45:32.102	SATISFIED_RULE : Encender las luces del baÃ±o cuando entra la abuela AUTOR: GrandMother (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-
19	346	9:09:31.0	10:45:42.659	SATISFIED_RULE : Luz en el espejo del baÃ±o al acercarse AUTOR: GrandMother (4b8961a8-66f7-4312-ad70-865b82c58ddc::Device-Locator::Locator-Service::Use
20	368	9:20:31.0	10:45:53.577	SATISFIED_RULE : Cerrar paso del agua para la abuela AUTOR: GrandMother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::User = Ju
21	374	9:20:31.0	10:45:53.624	SATISFIED_RULE : Si estoy en la ducha se apaga la luz del espejo AUTOR: Mother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::Use
22	386	9:29:31.0	10:46:02.678	SATISFIED_RULE : Si estoy en la ducha se apaga la luz del espejo AUTOR: Mother (b4165b67-96cf-49ca-afd9-0a2b17ec7756::Device-Locator::Locator-Service::Use
23	410	9:55:31.0	10:46:29.260	SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = N
24	420	9:55:31.0	10:46:29.260	SATISFIED_RULE : Apagar las luces del baÃ±o cuando no haya nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::U
25	482	10:33:31.0	10:47:07.903	SATISFIED_RULE : Por la maÃ±ana en la cocina al entrar TV 5 AUTOR: Mother (AND(5ad1a886-b369-437a-a3dc-00bfb61fbe8e::Device-Locator::Locator-Service::U
26	524	10:45:31.0	10:47:18.930	SATISFIED_RULE : Nerea en el baÃ±o luces y cerrar ventana AUTOR: Nerea (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = Ner
27	542	10:52:31.0	10:47:26.324	SATISFIED_RULE : Apagar Media en el baÃ±o si no hay nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::User = N
28	552	10:52:31.0	10:47:26.355	SATISFIED_RULE : Apagar las luces del baÃ±o cuando no haya nadie AUTOR: Admin (975abbc7-b0b8-4c1c-ba2b-ce1cf4805dff::Device-Locator::Locator-Service::U
29	570	10:56:31.0	10:47:30.490	SATISFIED_RULE : Apagar las luces de la sala cuando no haya nadie AUTOR: Admin (11fd5cba-3891-480e-a5d2-f87becb61f5f::Device-Locator::Locator-Service::Us
30	580	10:56:31.0	10:47:30.490	SATISFIED_RULE : Apagar Media en la sala si no hay nadie AUTOR: Admin (11fd5cba-3891-480e-a5d2-f87becb61f5f::Device-Locator::Locator-Service::User = Non
31	604	11:02:31.0	10:47:35.957	SATISFIED_RULE : Parar riego al detectar alguien en el jardin AUTOR: Admin (badb834d-16fe-484d-9443-447606161a42::Device-Locator::Locator-Service::User !=

Fig. E.5 Vista reglas satisfechas

E.6 Acciones ejecutadas

En la siguiente figura E.6 se muestran como ejemplo unas cuantas acciones ejecutadas.

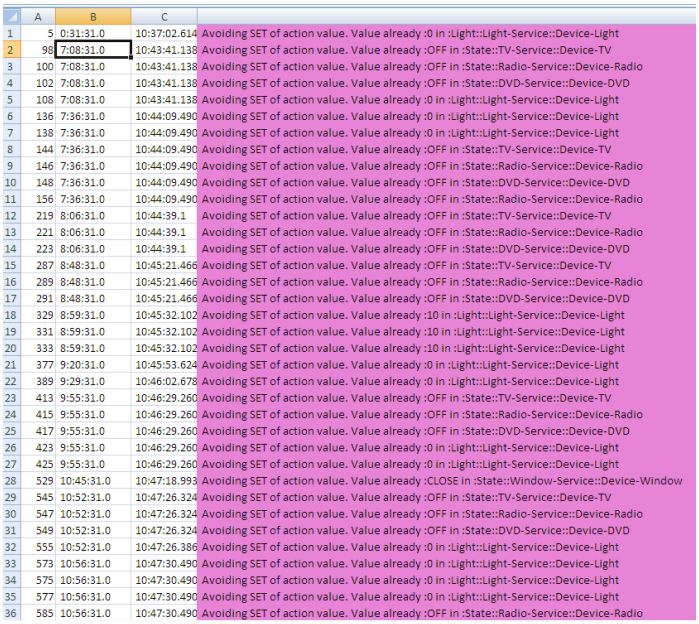


	A	B	C	D	E	F
1	12	5:00:31.0	10:41:33.398	Action executed = set_Irrigation_State : Device-Irrigation : Irrigation-Service	0,0076539729	24795978561954
2	38	6:15:31.0	10:42:48.319	Action executed = set_Light_Light : Device-Light : Light-Service	0,0071688726	25019627919165
3	66	6:48:31.0	10:43:21.715	Action executed = set_Light_Light : Device-Light : Light-Service	0,0029847300	25119309532311
4	90	7:04:31.0	10:43:37.488	Action executed = set_Light_Light : Device-Light : Light-Service	0,0020670761	25166427775569
5	110	7:08:31.0	10:43:41.138	Action executed = set_Light_Light : Device-Light : Light-Service	0,0033925832	25177308542046
6	134	7:36:31.0	10:44:09.490	Action executed = set_Light_Light : Device-Light : Light-Service	0,0028589843	25261943802201
7	170	7:41:31.0	10:44:14.252	Action executed = set_Light_Light : Device-Light : Light-Service	0,0029530402	25276166357229
8	172	7:41:31.0	10:44:14.252	Action executed = set_Light_Light : Device-Light : Light-Service	0,0040883183	25276166357229
9	188	7:47:31.0	10:44:20.611	Action executed = set_Light_Light : Device-Light : Light-Service	0,0018030676	25295147012340
10	229	8:06:31.0	10:44:39.32	Action executed = set_Light_Light : Device-Light : Light-Service	0,0316571528	25350056622117
11	231	8:06:31.0	10:44:39.32	Action executed = set_Light_Light : Device-Light : Light-Service	0,0331383270	25350056622117
12	269	8:29:31.0	10:45:02.465	Action executed = set_Louver_State : Device-Louver : Louver-Service	0,0030309279	25420084904799
13	297	8:48:31.0	10:45:21.466	Action executed = set_Light_Light : Device-Light : Light-Service	0,0052597462	25476808760613
14	299	8:48:31.0	10:45:21.466	Action executed = set_Light_Light : Device-Light : Light-Service	0,0066114304	25476808760613
15	301	8:48:31.0	10:45:21.466	Action executed = set_Light_Light : Device-Light : Light-Service	0,0077588467	25476808760613
16	353	9:09:31.0	10:45:42.659	Action executed = set_Light_Light : Device-Light : Light-Service	0,0023016053	25540100090424
17	371	9:20:31.0	10:45:53.624	Action executed = set_Irrigation_State : Device-Irrigation : Irrigation-Service	0,0698288080	25572623773797
18	485	10:33:31.0	10:47:07.903	Action executed = set_TV_Channel : Device-TV : TV-Service	0,0040054201	25794557398674
19	487	10:33:31.0	10:47:07.918	Action executed = set_TV_State : Device-TV : TV-Service	0,0057502510	25794557398674
20	527	10:45:31.0	10:47:18.993	Action executed = set_Light_Light : Device-Light : Light-Service	0,0745223573	25827438575250
21	557	10:52:31.0	10:47:26.386	Action executed = set_Light_Light : Device-Light : Light-Service	0,0649616490	25849541922570
22	583	10:56:31.0	10:47:30.490	Action executed = set_TV_State : Device-TV : TV-Service	0,0036044661	25861977253557
23	607	11:02:31.0	10:47:35.957	Action executed = set_Irrigation_State : Device-Irrigation : Irrigation-Service	0,0019909177	25878265830135
24	613	11:02:31.0	10:47:36.35	Action executed = set_Play_TTS : Device-Media : Media-Service	0,0884614702	25878265830135
25	643	11:15:31.0	10:47:49.334	Action executed = set_Heater_Temperature : Device-Heater : Heater-Service	0,0041522422	25918214971968
26	701	11:44:31.0	10:48:18.359	Action executed = set_TV_State : Device-TV : TV-Service	0,0074879901	26004880478511
27	707	11:44:31.0	10:48:18.375	Action executed = set_TV_Channel : Device-TV : TV-Service	0,0088991232	26004880478511
28	739	11:53:31.0	10:48:27.288	Action executed = set_TV_State : Device-TV : TV-Service	0,0475880556	26031405948930
29	805	12:19:31.0	10:49:20.921	Action executed = set_Light_Light : Device-Light : Light-Service	0,0033672512	26191630223604
30	807	12:19:31.0	10:49:20.921	Action executed = set_Light_Light : Device-Light : Light-Service	0,0047064273	26191630223604
31	869	12:37:31.0	10:49:39.311	Action executed = set_TV_State : Device-TV : TV-Service	0,0037682744	26246526091488

Fig. E.6 Vista acciones ejecutadas

E.7 Acciones no ejecutadas

En la siguiente figura E.7 se muestran como ejemplo unas cuantas acciones no ejecutadas por tener de antemano el valor deseado.



	A	B	C	
1	5	0:31:31.0	10:37:02.614	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
2	98	7:08:31.0	10:43:41.138	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Device-TV
3	100	7:08:31.0	10:43:41.138	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio
4	102	7:08:31.0	10:43:41.138	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Device-DVD
5	108	7:08:31.0	10:43:41.138	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
6	136	7:36:31.0	10:44:09.490	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
7	138	7:36:31.0	10:44:09.490	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
8	144	7:36:31.0	10:44:09.490	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Device-TV
9	146	7:36:31.0	10:44:09.490	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio
10	148	7:36:31.0	10:44:09.490	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Device-DVD
11	156	7:36:31.0	10:44:09.490	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio
12	219	8:06:31.0	10:44:39.1	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Device-TV
13	221	8:06:31.0	10:44:39.1	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio
14	223	8:06:31.0	10:44:39.1	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Device-DVD
15	287	8:48:31.0	10:45:21.466	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Device-TV
16	289	8:48:31.0	10:45:21.466	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio
17	291	8:48:31.0	10:45:21.466	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Device-DVD
18	329	8:59:31.0	10:45:32.102	Avoiding SET of action value. Value already :10 in :Light::Light-Service::Device-Light
19	331	8:59:31.0	10:45:32.102	Avoiding SET of action value. Value already :10 in :Light::Light-Service::Device-Light
20	333	8:59:31.0	10:45:32.102	Avoiding SET of action value. Value already :10 in :Light::Light-Service::Device-Light
21	377	9:20:31.0	10:45:53.624	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
22	389	9:29:31.0	10:46:02.678	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
23	413	9:55:31.0	10:46:29.260	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Device-TV
24	415	9:55:31.0	10:46:29.260	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio
25	417	9:55:31.0	10:46:29.260	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Device-DVD
26	423	9:55:31.0	10:46:29.260	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
27	425	9:55:31.0	10:46:29.260	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
28	529	10:45:31.0	10:47:18.993	Avoiding SET of action value. Value already :CLOSE in :State::Window-Service::Device-Window
29	545	10:52:31.0	10:47:26.324	Avoiding SET of action value. Value already :OFF in :State::TV-Service::Device-TV
30	547	10:52:31.0	10:47:26.324	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio
31	549	10:52:31.0	10:47:26.324	Avoiding SET of action value. Value already :OFF in :State::DVD-Service::Device-DVD
32	555	10:52:31.0	10:47:26.386	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
33	573	10:56:31.0	10:47:30.490	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
34	575	10:56:31.0	10:47:30.490	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
35	577	10:56:31.0	10:47:30.490	Avoiding SET of action value. Value already :0 in :Light::Light-Service::Device-Light
36	585	10:56:31.0	10:47:30.490	Avoiding SET of action value. Value already :OFF in :State::Radio-Service::Device-Radio

Fig. E.7 Vista acciones no ejecutadas

E.8 Conflictos detectados

En la siguiente figura E.8 se muestran como ejemplo los conflictos detectados y la forma en que se han resuelto. Por ejemplo, en la línea 6 de la figura E.8, se puede ver como una regla se antepone a la otra. Al cumplirse las condiciones de la regla de sistema, el valor de la acción original es modificado.

	A	B	C	D	E
1	349	9:09:31.0	10:45:42.659	Starting to evaluate Afected_OTHER_USER Rule: Encender luz del espejo ->set_Light_Light = 10	0,0004126170
2	352	9:09:31.0	10:45:42.659	NOT_Afected_OTHER_USER_Rule	0,0005032895
3	635	11:15:31.0	10:47:49.334	Starting to evaluate Afected_OTHER_USER Rule: Subir TÃª casa si baja la media de 20Â°C ->set_Heater_Temperature = 22	0,0020715813
4	638	11:15:31.0	10:47:49.334	NOT_Afected_OTHER_USER_Rule	0,0021334843
5	639	11:15:31.0	10:47:49.334	Starting to evaluate AFECTED_MANDATORY_RULE: set_Heater_Temperature = 22	0,0021506750
6	642	11:15:31.0	10:47:49.334	Afected: Applying_Mandatory_Rule over User's one.set_Heater_Temperature = 24	0,0024261570
7	645	11:15:31.0	10:47:49.334	Starting to evaluate Afected_OTHER_USER Rule: TV canal 2 entre las 15:00 y las 16:00 al sentarme en el sofa ->set_TV_State = ON	0,0042117001
8	648	11:15:31.0	10:47:49.334	NOT_Afected_OTHER_USER_Rule	0,0044134809
9	651	11:15:31.0	10:47:49.334	Starting to evaluate Afected_OTHER_USER Rule: TV canal 2 entre las 15:00 y las 16:00 al sentarme en el sofa ->set_TV_Channel = 8	0,0044858889
10	654	11:15:31.0	10:47:49.334	NOT_Afected_OTHER_USER_Rule	0,0046654745
11	697	11:44:31.0	10:48:18.359	Starting to evaluate Afected_OTHER_USER Rule: Apagar Media en la cocina si no hay nadie ->set_TV_State = ON	0,0051923095
12	700	11:44:31.0	10:48:18.359	NOT_Afected_OTHER_USER_Rule	0,0055222179
13	703	11:44:31.0	10:48:18.359	Starting to evaluate Afected_OTHER_USER Rule: Por la maÃ±ana en la cocina al entrar TV 5 ->set_TV_Channel = 10	0,0075777423
14	706	11:44:31.0	10:48:18.375	NOT_Afected_OTHER_USER_Rule	0,0079791144
15	809	12:19:31.0	10:49:20.921	Starting to evaluate Afected_OTHER_USER Rule: Nerea luces por el pasillo por la noche ->set_Light_Light = 10	0,0048294843
16	812	12:19:31.0	10:49:20.968	NOT_Afected_OTHER_USER_Rule	0,0475083454

Fig. E.8 Vista reglas en conflicto

E.9 Resumen de datos

En la siguiente figura E.9 se muestran los datos que resumen el test realizado a lo largo de un día.

21	Número de eventos recibidos	153
22	Número de reglas evaluadas	222
23	Reglas evaluadas y no ejecutadas	175
24	Reglas evaluadas y ejecutadas	47
25	Acciones para ejecutar	93
26	Acciones ejecutadas	32
27	Ejecución de acciones descartadas (mismo valor)	61
28	Reglas en conficto	8
29	Reglas en conficto que se ejecuta otra regla de mayor prioridad	1

Fig. E.9 Vista datos resumen

Destacar que partiendo de 79 acciones simuladas, se han recibido 153 eventos. La diferencia se encuentra en los eventos producidos por las acciones que el motor de reglas ha ejecutado. De todas formas, todavía tenemos que 79 + 32 no llega a 153. En este caso, la explicación está en que los dispositivos de luz, para pasar de encendido a apagado, pasan por estados intermedios. No son dispositivos ‘todo o nada’ si no que simulan un variador de intensidad de luz, por lo que para pasar de 0 (apagado) a 10 (encendido) pasan por los valores intermedios de 2,4,6 y 8, siendo estos cambios también recibidos como eventos en el motor de reglas.

Los datos anteriores solo son representativos para la prueba realizada con un número concreto de dispositivos virtuales, una secuencia de simulación concreta y para un número de reglas concreto. Toda modificación en alguno de los elementos anteriormente mencionados generaría unos valores finales totalmente diferentes. No obstante, después de repetir la misma secuencia de test cinco veces, partiendo de la misma situación inicial, los valores obtenidos han sido los mismos.

Anexo F: Creación de Reglas utilizando TAMAmI

En este anexo se describe el proceso de creación de varias reglas utilizando como ejemplo varios casos de uso.

Se explicarán diversas reglas significativas para tratar de explicar la dinámica y filosofía de creación utilizada.

- **Regla 1:** queremos que todas las televisiones, radios y DVDs se apaguen cuando nadie se encuentre en la habitación correspondiente. Con esta regla se pretende analizar la creación múltiple de reglas.

  Para crear esta regla lo más fácil es crear la misma regla por cada estancia en el hogar. Si elegimos la cocina por ejemplo: en parte condicional se trata de detectar que no haya nadie en la estancia, y en la parte de acciones no hay más que indicar cuál es el dispositivo que dispone del servicio de televisión, radio o DVD y apagarlo. La figura F.1 muestra la creación de la regla en la habitación de la abuela y se describe la acción que apaga la radio.

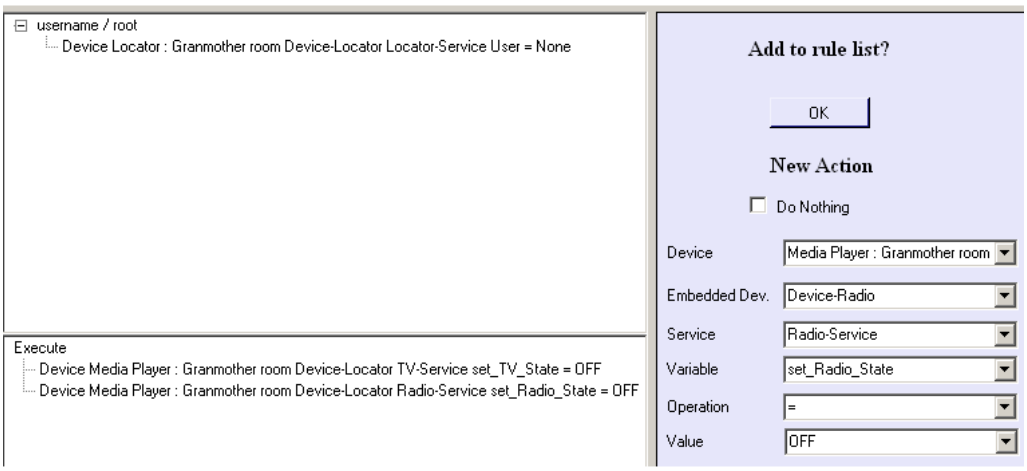


Fig. F.1 Vista regla1

- **Regla 2:** a partir de las once de la noche y hasta las siete de la mañana no se podrá poner el volumen de la televisión y de la radio de la sala más alto que la mitad de lo posible. Lo mismo entre las tres de la tarde y las cuatro de la tarde. Con esta regla se pretende analizar las condiciones de tipo tiempo para crear periodos de tiempo.

  Es decir, siempre que alguien intente cambiar el volumen de audio de algún dispositivo de la sala en la franja horaria dada se verificará si es más alto de lo establecido por esta regla. Por otro lado, esta regla va a estar compuesta por varias condiciones, unas de tipo tiempo y otras compuestas. En la figura F.2, se ve como la parte condicional es una combinación de condiciones de tipo tiempo y condiciones compuestas.

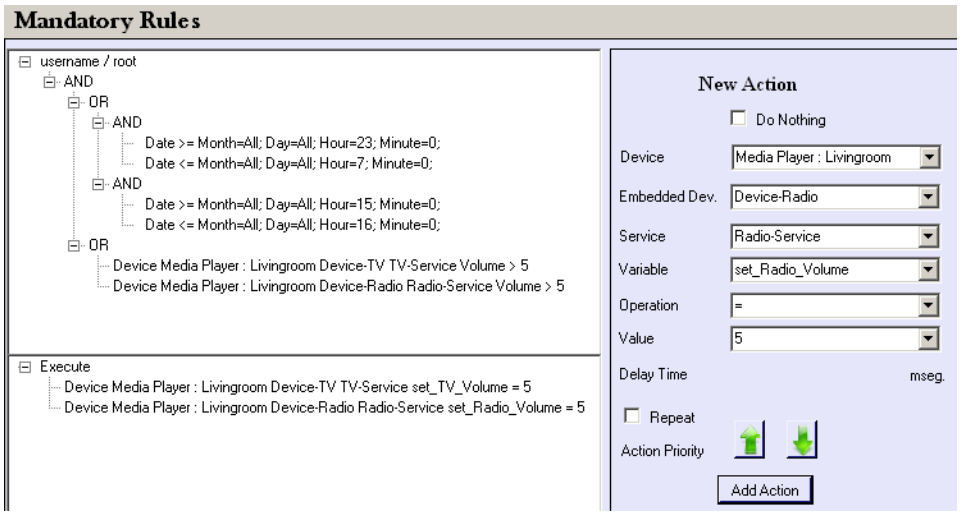


Fig. F.2 Vista de la regla para limitar el volumen de los dispositivos de audio entre las 11 y las 7 de la mañana y entre las 3 y las 4 de la tarde

- **Regla 4:** nadie podrá poner una temperatura mayor que 23 ºC en el termostato de la sala. Con esta regla se pretende analizar la creación de reglas de sistema.

  Esta regla es una regla del sistema. Se deberá probar siempre que se intente modificar la temperatura del termostato de la sala. Su creación se muestra en la figura F.3.

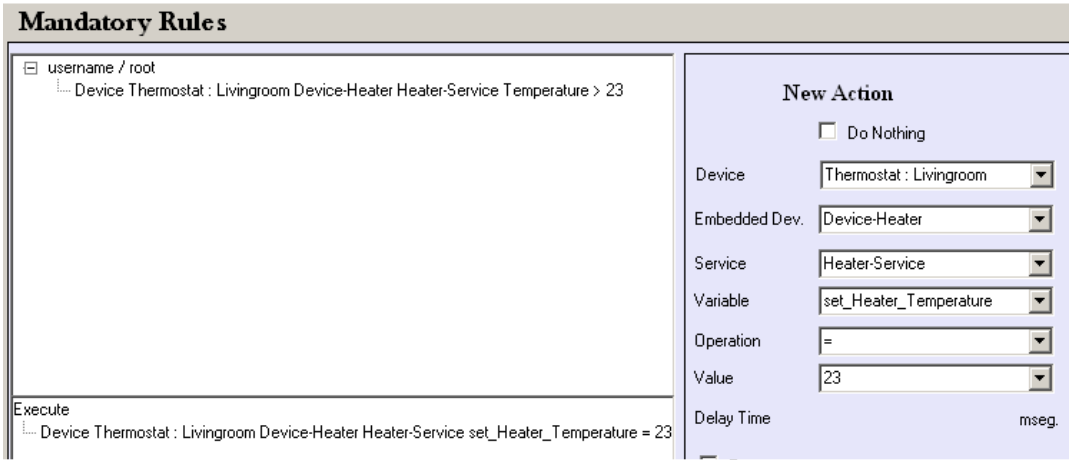


Fig. F.3 Vista de la regla que establece que nadie pueda poner una temperatura mayor que 23ºC en el termostato de la sala

- **Regla 4:** nadie podrá poner una temperatura menor que 22ºC en el termostato de la sala en los meses de junio-julio-agosto. Con esta regla se pretende analizar la creación de reglas de sistema con condiciones de tiempo y ordinarias.

  En este caso, la regla es una extensión de la anterior, ya que incorpora dos condiciones de tiempo a la acción de la regla anterior. En la figura F.4 se puede observar la diferencia entre las dos.

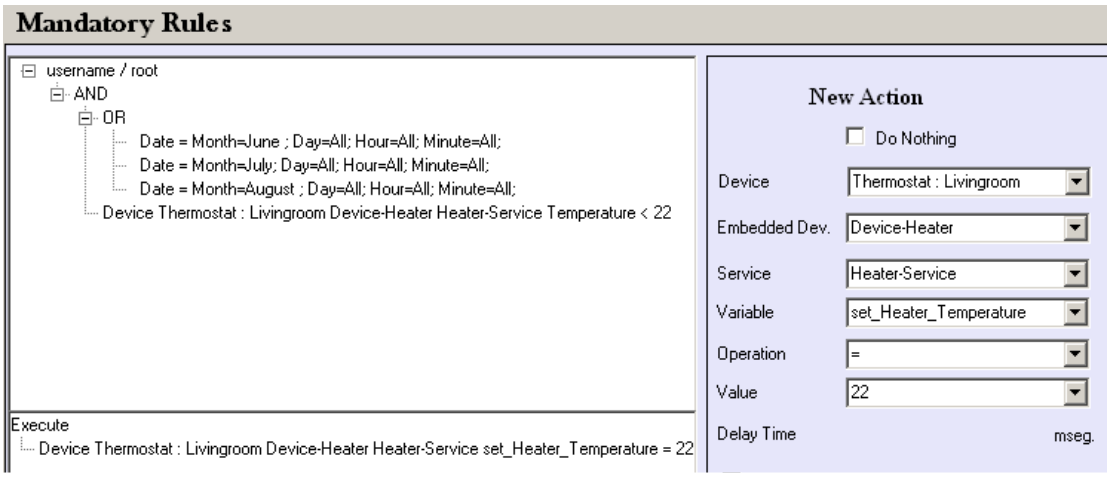


Fig. F.4 Vista de la regla que establece que nadie pueda poner una temperatura menor que 22º en la sala en los meses de junio, julio y agosto

- **Regla 5**: si se detecta a alguien en el jardín por la noche, que se encienden las luces.

  Esta regla es parecida a las anteriores, y se podría complementar con una segunda regla adicional para apagar las luces si no se encuentra nadie en el jardín. Justo la acción contraria para dejar en un estado coherente el sistema. Las condiciones y la acción de esta regla se muestran en la figura F.5.

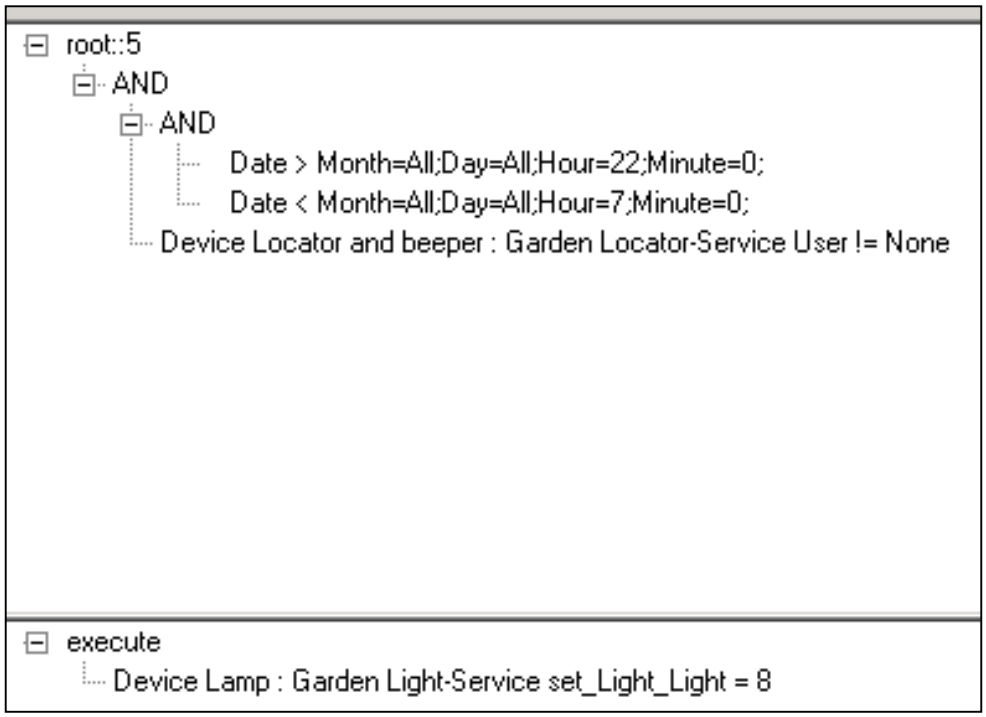


Fig. F.5 Vista de la regla para encender las luces cuando haya alguien en el jardín

La regla para apagar las luces en caso de que estén encendidas y no haya nadie en el jardín queda como se muestra en la figura F.6.

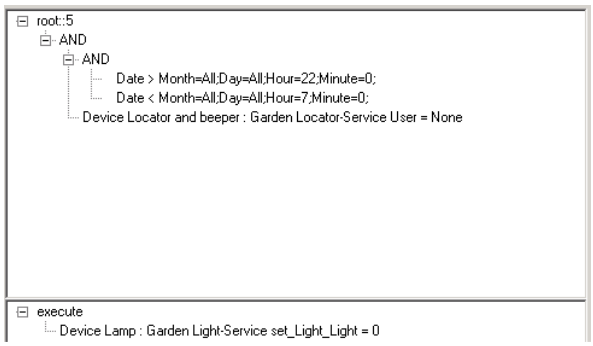


Fig. F.6 Vista de la regla para apagar las luces en el jardín cuando no haya nadie

Como era de esperar, el ER alerta de un posible conflicto entre las dos reglas que acabamos de crear (Fig. F.7), ya que una de ellas intenta poner a cero (apagar) las luces y la otra intenta ponerlas al valor máximo -10- (encender).

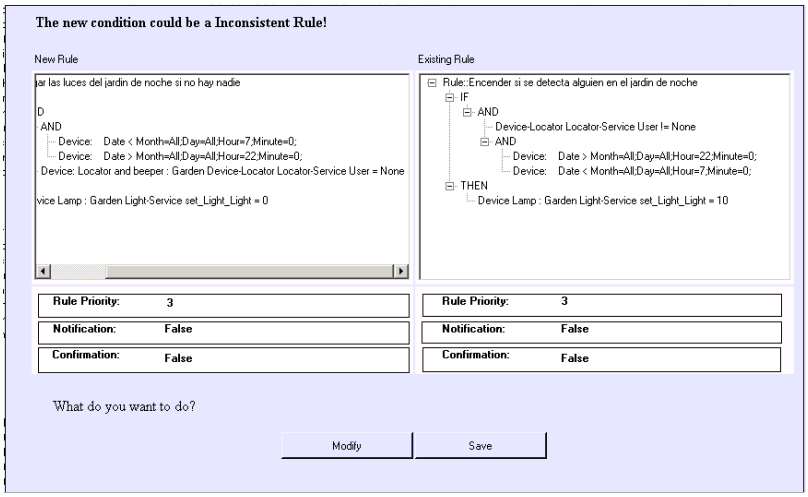


Fig. F.7 Vista de la detección de conflicto entre dos reglas

Siendo este el caso en el que la situación está controlada y que las acciones son correctas, guardaremos ambas reglas.

- **Regla 6:** si hay alguien en el jardín que se paren los aspersores.

  La figura F.8 muestra la regla en el ER.

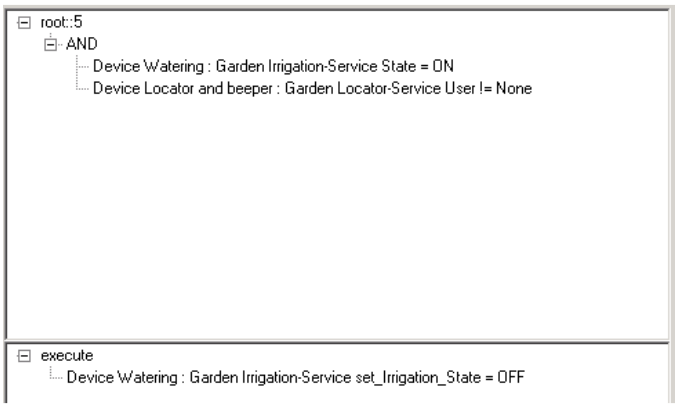


Fig. F.8 Vista de la regla que se paren los aspersores cuando alguien está en el jardín

- **Regla 7:** si alguien está en el pasillo, que se encienden las luces.

  Tal y como ha ocurrido en la regla 1, en este caso también se puede abordar la creación de la regla utilizando otra regla de tipo grupo. En este caso, primero debemos crear la regla de tipo grupo para agrupar los dispositivos que ofrezcan luz y estén situados en el pasillo. Posteriormente, crearemos la regla para asociar el grupo de dispositivos con la condición que la hace cumplir (Fig. F.9).

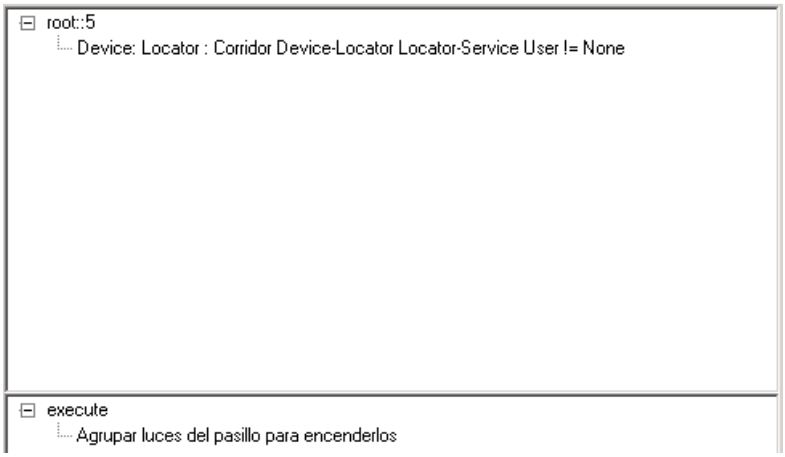


Fig. F.9 Vista de la regla 'Si hay alguien en el pasillo, que se enciendan las luces'

- **Regla 8**: si la temperatura media en la sala, cocina y pasillo es menor de 20ºC, poner la temperatura a 24ºC en las habitaciones involucradas.

  Esta regla está pensada para crear una condición calculada. La regla se queda como se muestra en la figura F.10.

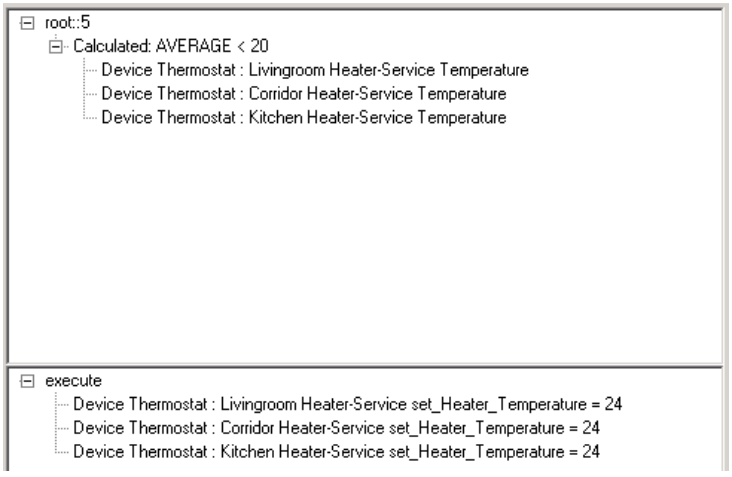


Fig. F.10 Vista de la regla que 'si la temperatura media en la sala, cocina y pasillo es menor de 20ºC se ponga la temperatura a 24ºC en ellas

Por otro lado, en cuanto a la acción de poner la temperatura a 24ºC, va a producir un conflicto con la regla 4 del sistema, con lo que el MR deberá detectar un conflicto y solucionarlo dependiendo de las prioridades asignadas. Este puede ser el caso en el

que en la vida real, el administrador del sistema establece unos límites, y los usuarios normales, sin tener conocimiento de ellos intentan realizar otras acciones. En este caso el motor de reglas será el encargado de ajustar las preferencias del usuario normal a los límites establecidos por el administrador.

Anexo G: Publicaciones del autor

En este anexo se detallan las publicaciones presentadas en foros nacionales e internacionales relacionadas con el autor.

[Jimeno+04] Jimeno R., Salvador Z., Lafuente A., Larrea M., Uribarren A., « An Architecture for the Personalized Control of Domotic Resources », European Symposium on Ambient Intelligence, EUSAI 2004.

[Uribarren+05] Uribarren A, Parra J., Uribe J.P., Zamalloa M., Makibar K., « Middleware para servicios distribuidos y aplicaciones móviles », Congreso Español de Informática, CEDI - UCAMI 2005.

[Parra+06] Parra J., Uribarren A, Uribe J.P., Makibar K., Olalde I., « Despliegue de servicios y aplicaciones en plataformas heterogéneas », Congreso Español de Informática, CEDI - CICU 2006.

[Uribarren+06a] Uribarren A, Parra J., Uribe J.P., Zamalloa M., Makibar K., « Middleware for Distributed Services and Mobile Applications », International Conference on Integrated Internet Ad Hoc and Sensor Networks, InterSense 2006.

[Uribarren+06b] Uribarren A, Parra J., Uribe J.P., Makibar K., Olalde I., Herrasti N., « Service Oriented Pervasive Applications Based On Interoperable Middleware », Workshop on Requirements and Solutions for Pervasive Software Infrastructures, RSPSI 2006.

[Uribarren+07a] Uribarren A, Parra J., Ali S., « Applications of Ambient Intelligence in Medical Devices and Clinical Environments », International Conference on e-Medical Systems: e-Medisys'07.

[Uribarren+07b] Uribarren A, Parra J., Uribe J.P., « Entorno AmI en sistemas de biodiagnóstico », Symposium of Ubiquitous Computing and Ambient Intelligence, CEDI - UCAMI 2007.

[Uribarren+08] Uribarren A., Parra J., Iglesias R., Uribe J.P., López-de-Ipiña D., « A Middleware Platform for Application Configuration, Adaptation and Interoperability », Workshop on Pervasive Adaptation, SASO - Perada 2008.

Otras publicaciones:

[Urbieta+07] Urbieta A., Barrutieta G., Parra J., Uribarren A., « Estado del arte de composición dinámica de servicios en entornos de computación ubicua », Symposium of Ubiquitous Computing and Ambient Intelligence , CEDI - UCAMI 2007.

[Urbieta+08] Urbieta A., Barrutieta G., Parra J., Uribarren A., « A Survey of Dynamic Service Composition Approaches for Ambient Systems », First Workshop on Software Organisation and MonIToring of Ambient Systems. SOMITAS 2008.

[Parra+09] Parra J., Anwar M., Uribarren A., Jacob E., El Saddik A., “Flexible Smart Home Architecture using Device Profile for Web Services: a Peer-to-Peer Approach”, International Journal of Smart Home. Vol.3, No.2, April, 2009. 39.