Generación automática de interfaces de usuario para el hogar digital. Caso de uso: Accesibilidad de personas con discapacidad visual.

José Ramón Padilla López
Universidad de Alicante
jpadilla@dtic.ua.es

Francisco Flórez Revuelta
Universidad de Alicante
florez@dtic.ua.es

Resumen

Este artículo presenta un nuevo lenguaje para describir interfaces de usuario, con un alto nivel de abstracción y basadas en una representación textual, para el desarrollo de aplicaciones del hogar digital. El objetivo de este lenguaje es describir la funcionalidad y servicios ofrecidos por el hogar facilitando el desarrollo de aplicaciones para la vida asistida por el entorno. Utilizando este lenguaje se ha desarrollado una aplicación móvil capaz de controlar diferentes aspectos de la vivienda: iluminación, control de puertas y persianas, aire acondicionado y calefacción, etcétera; la cual persigue aumentar la autonomía de las personas dentro de sus hogares. Esta aplicación es presentada en este artículo como caso de estudio.

Abstract

This paper introduces a new text-based user interface description language, with a high degree of abstraction, designed for the development of digital home applications. The main goal of the language is to describe all functionalities and services offered by a digital home, providing an easy-way to develop ambient assisted living applications. Using this language, we have developed a mobile application which is able to control several home features like: lighting, control of doors and blinds; air conditioning, heating, and so forth. This application allows increasing the personal autonomy of somebody living in his home. In this paper this application is presented as a study case.

1. Introducción

Las interfaces de usuario (UI) son una cuestión crucial en el desarrollo de las Tecnologías de la Información, las Comunicaciones y el Control (TICC). En ellas se recogen toda la interacción que ocurre entre el usuario y el computador, por lo que conceptos como: la eficacia de la interfaz para llevar a cabo operaciones, la usabilidad, o la retroalimentación en forma de información útil que ésta aporta al usuario; son de vital importancia. Estos conceptos cobran todavía más fuerza en aplicaciones del campo de la Vida Asistida por el Entorno (AAL). En estos entornos las aplicaciones deben ser accesibles y usables para todas las personas, por lo que se debe tener en cuenta desde el primer momento el perfil de cada tipo de usuario, considerando sus capacidades para acceder a tecnologías y servicios. El desarrollo de este tipo de aplicaciones se puede englobar dentro del paradigma del Diseño Universal [1]. Sin embargo, los principios del diseño universal todavía no se aplican de forma generalizada. Prueba de ello es que la interfaz de usuario gráfica (GUI) es la más común en el desarrollo de aplicaciones en la actualidad. Como puede intuirse, debido a su naturaleza gráfica, las personas con discapacidad visual no pueden manejar adecuadamente este tipo de interfaces.

Para hacer frente a este problema existen diferentes aproximaciones. La más utilizada consiste en el empleo de lectores de pantalla que leen los elementos gráficos presentes en la interfaz de usuario, como por ejemplo, botones, etiquetas y cuadros de texto, entre otros. Otro modo consiste en enriquecer los elementos de la interfaz gráfica con información auditiva o táctil para poder representar esos elementos en otros dispositivos no gráficos [2]. Por último, en lugar de intentar añadir características a las GUIs para que sean accesibles, el paradigma de interfaz de usuario dual contempla el diseño de dos tipos de UI, visuales y no visuales, que se adapten a las características del usuario [3], aumentando de esta manera la usabilidad.

Para el desarrollo del presente trabajo se ha decidido seguir el paradigma de interfaz de usuario dual. Sin embargo, el desarrollo de una UI específica para cada modalidad es una tarea que demanda muchos recursos, esto sin considerar la cuestión de la portabilidad. En la actualidad existen una gran cantidad de plataformas, tanto móviles como de escritorio, para las cuales se pueden desarrollar aplicaciones. Las diferencias visibles entre cada una de ellas obliga a desarrollar la interfaz de usuario en cada plataforma soportada por la aplicación, provocando que el desarrollo de la UI se haya convertido en una tarea que consume muchos recursos. Debido a esto, se están dedicando importantes esfuerzos a la creación de lenguajes de descripción de interfaces de usuario (UIDL) que capturen la esencia de las UI. Por medio de estos lenguajes es posible independizar la interfaz de usuario de la plataforma y de la modalidad: gráfica, textual, voz, etcétera; facilitando la portabilidad de la aplicación y la generación de diferentes tipos de UI con la consiguiente reducción tanto en tiempo como en esfuerzo durante la fase de desarrollo [4,5].

El lenguaje que se presenta en este trabajo está diseñado para describir interfaces de usuario con un alto nivel de abstracción, teniendo en cuenta las cuestiones mencionadas y los requisitos impuestos por el entorno desde etapas tempranas del diseño. El uso hacia el que se enfoca este UIDL ha marcado también el diseño del mismo. Principalmente, las aplicaciones que hagan uso de este lenguaje serán aquellas que actúen como interfaz remota de los servicios disponibles dentro de un hogar inteligente. En ese sentido, se ha desarrollado una aplicación móvil para personas ciegas que, utilizando una interfaz descrita con el lenguaje que se presenta en este trabajo, permite controlar todos los servicios del hogar. El trabajo presentado en este artículo se estructura de la siguiente manera: en la sección 2 se realiza una revisión del estado de la cuestión sobre UIDL y generación automática de interfaces de usuario. En la sección 3 se introducen los principios de diseño considerados a la hora de diseñar el UIDL. La especificación del lenguaje se encuentra en la sección 4. En la sección 5 se presenta un caso de estudio sobre el desarrollo de una interfaz para controlar los servicios del hogar inteligente, accesible para personas ciegas. En la sección 6 se presenta el prototipo de hogar digital donde se ha validado la interfaz. Por último, en la sección 7 se concluye el presente trabajo destacando los principales aspectos del mismo y los trabajos futuros.

2. Trabajos relacionados

La idea de definir la interfaz de usuario mediante una descripción de la misma se remonta a mediados de la década de los 80, más concretamente, al concepto de Sistemas de Gestión de Interfaces de Usuario (UIMS) [6]. El objetivo de estos sistemas es la separación de la UI de la lógica de la aplicación utilizando modelos. Sin embargo, la necesidad de diseñar UI independientes del dispositivo y automatizar, en parte, el proceso de implementación, ha dado paso a los Lenguajes de Descripción de Interfaces de Usuario (UIDL).

Mediante estos lenguajes se define la interfaz de usuario de forma descriptiva. Posteriormente esta descripción es procesada en tiempo de desarrollo o ejecución para producir la interfaz de usuario. Las descripciones que ofrecen estos lenguajes consideran aspectos como la independencia de la plataforma, la independencia de la modalidad y el nivel de abstracción. En [7] se presenta un marco que estructura el ciclo de vida de las UI sensibles al contexto en cuatro niveles de abstracción: i) tareas y conceptos, ii) interfaz de usuario abstracta (AUI), iii) interfaz de usuario concreta (CUI) y iv) interfaz de usuario final (FUI). Así, la AUI es independiente de la plataforma y la modalidad, la CUI es independiente de la plataforma y la FUI es dependiente de la plataforma y la modalidad. Esta clasificación ha sido tenida en cuenta en los desarrollos posteriores de muchos UIDL, por lo que es posible encontrar en sus especificaciones estos niveles de abstracción de forma implícita o explícita. Muchos de los lenguajes UIDL basan su sintaxis en XML debido a la facilidad de edición, soporte software y multitud de analizadores y procesadores disponibles para este lenguaje. Son muchos los lenguajes que existen para describir interfaces de usuario. A continuación se presenta una visión general de aquellos UIDL basados en XML que consideramos más relevantes por su frecuente aparición en la literatura [4,5].

UIML [8] es un UIDL que tiene como propósito el soportar el desarrollo de UIs para varias plataformas utilizando una descripción independiente de la misma. Permite describir la apariencia de la aplicación, la interacción y la conexión de la UI con la lógica de la aplicación. DISL [9] es una extensión de UIML, un subconjunto que extiende este último, para habilitar la descripción de diálogos independientes de la modalidad.

XIML [10,11] es un lenguaje de propósito general para almacenar y manipular datos procedentes de interacciones. Puede almacenar muchos tipos de modelos de UI, al igual que las relaciones existentes entre los modelos y los datos. Este lenguaje se compone principalmente de cuatro tipos de componentes: modelos, elementos, atributos y relaciones entre los elementos.

UsiXML [12] se estructura de acuerdo a los cuatro niveles de abstracción definidos en [7], siendo

capaz de moverse entre ellos hasta llegar a la FUI mediante transformaciones. El objetivo de UsiXML es soportar todas las características de los UIDL desarrollados antes que él y, por tanto, cuenta con muchas de ellas. Permite la especificación de muchos tipos de modelos de UIs como: tareas, dominios, presentaciones y contextos de uso; con un soporte importante para describir relaciones entre todos los modelos soportados.

WSXL [13] es un lenguaje destinado al desarrollo de interfaces de usuario web para aplicaciones de Internet. Los dos objetivos principales de WSXL son: ofrecer un modelo de construcción de aplicaciones web que funcionen a través de una amplia variedad de canales; permitir la creación de aplicaciones web a partir de otras ya creadas. Este lenguaje se basa en estándares abiertos ya establecidos y es independiente de la plataforma, del navegador web y de los lenguajes de presentación.

Los UIDL revisados pueden describir el tipo de UI requerida por las aplicaciones que acceden remotamente a los servicios de un hogar inteligente, pero continúan añadiendo complejidad al lenguaje.

Además de los UIDL mostrados, se han considerado también otros lenguajes desarrollados por compañías y organizaciones de software que están bastante extendidos en la actualidad: MXML (Adobe) [14], XAML (Microsoft) [15], XUL (Mozilla) [16] y QML (Nokia) [17]. MXML es un lenguaje declarativo empleado en aplicaciones Adobe Flex para describir interfaces gráficas y su comportamiento. XAML es empleado en aplicaciones .NET para describir los elementos gráficos que se muestran por pantalla, así como los cambios de estado de éstos debido a la interacción con el usuario. XUL es un lenguaje, basado en los estándares web existentes, empleado en muchas aplicaciones de Mozilla, entre ellas el conocido navegador web Firefox. Finalmente, QML es un lenguaje basado en JavaScript para describir interfaces de usuario gráficas que forma parte del framework Qt de desarrollo de aplicaciones multi-plataforma. No obstante, estos lenguajes tienen un fuerte acoplamiento con sus propias plataformas y, además, están diseñados para describir, principalmente, interfaces de usuario gráficas.

Podemos encontrar también en la literatura otros trabajos más alineados con el tema del presente trabajo. En [18] se presenta el concepto de Consola Remota Universal (URC), como un dispositivo capaz de acceder a otros dispositivos o servicios compatibles para manejarlos a través de una UI generada de forma automática. Para llevar a cabo este proyecto proponen AAIML, un UIDL que permite describir interfaces independientes de la modalidad. El URC fue adoptado como estándar en 2008 por el organismo ISO/IEC JTC1 y publicado como ISO/IEC 24752. Otro trabajo similar es el presentado en [19], donde se introduce el concepto de Controlador Universal Personal (PUC), que guarda mucha relación con URC. En [20] se presenta el lenguaje UIDL que utiliza PUC para describir las UI de acceso remoto a los dispositivos.

3. Principios de diseño

Durante el diseño de la especificación de nuestro UIDL se han tenido en cuenta los siguientes requisitos y principios de diseño. Por un lado, puesto que el lenguaje debe ser interpretado por las aplicaciones, se hace necesario el mantener una sintaxis fácil de analizar e interpretar. Por otro lado, el mismo lenguaje debe ser fácil de manipular, sin necesidad de recurrir a aplicaciones específicas. Estos requisitos han motivado el uso de una sintaxis basada en XML, la cual puede ser manipulada con cualquier editor de textos, contando además con un buen soporte para su procesamiento por parte de los lenguajes de programación de alto nivel.

Otro aspecto que se ha tenido en cuenta ha sido el de no incluir información alguna sobre la posición de los elementos dentro de la UI ni de la modalidad del tipo de interfaz. El objetivo es diseñar un lenguaje que se centre única y exclusivamente en la descripción de la funcionalidad que puede encontrarse en los servicios disponibles dentro del hogar inteligente. La modalidad de la interfaz será determinada en tiempo de ejecución por la aplicación encargada de interpretar la descripción, a la que denominamos intérprete. Por tanto, en lugar de sobrecargar al lenguaje añadiéndole complejidad, se ha optado por trasladar esa sobrecarga al intérprete. De esta manera, una misma descripción puede instanciarse como una interfaz basada en voz o una interfaz gráfica dependiendo del soporte ofrecido por el intérprete.

De lo anterior se deriva otro principio de diseño, la independencia del lenguaje sobre la plataforma. En la especificación del lenguaje no se encuentran, por tanto, elementos específicos de ninguna plataforma. Sin embargo, estos pueden ser deducidos por el intérprete a partir de la descripción. A pesar de tener que implementar un intérprete para cada modalidad de interfaz y plataforma, este enfoque añade mayor flexibilidad, ya que las descripciones de las UIs ya definidas no necesitan modificarse.

A la hora de describir la funcionalidad se ha optado por mantener un alto nivel de abstracción que sea independiente de la presentación. Por ese motivo, no se emplean abstracciones de elementos como botones, cuadros de texto, etiquetas, etc. El tipo de aplicaciones hacia el que va dirigido este lenguaje son aquellas que actúan como interfaz de acceso remoto a los servicios de un hogar digital. Estos servicios son: control de iluminación; apertura

y cierre de puertas, ventanas y persianas; control de temperatura; activación de escenarios; consultas del estado de ciertos dispositivos; etc. La funcionalidad que encontramos en este tipo de servicios puede modelarse como entradas y salidas de datos, o en otras palabras, como acciones del usuario que provocan respuestas por parte de los servicios del hogar. Por tanto, uno de los principios de diseño ha sido el poder describir toda la funcionalidad de los diferentes servicios disponibles en forma de acciones y respuestas.

A pesar del alto nivel de abstracción desde el punto de vista de la UI, al describir la funcionalidad en forma de acciones y respuestas existe una relación directa con la funcionalidad real. Debido a esto, uno de los requisitos del lenguaje es el de poder asociar a cada acción la funcionalidad real sobre la que se va a actuar para ejecutar la acción. El intérprete es el encargado de establecer los mecanismos de comunicación necesarios entre la UI y la funcionalidad.

4. Especificación del lenguaje

El UIDL que presentamos en este trabajo está basado en XML y permite describir la funcionalidad que se encuentra en los servicios del hogar en forma de acciones y respuestas. Este lenguaje contiene la suficiente capacidad descriptiva como para generar automáticamente la UI de acceso a los servicios sin incluir información sobre la apariencia de la misma. La especificación del lenguaje se divide en dos apartados: la descripción de la funcionalidad y la descripción de la UI. Esta separación se encuentra de forma explícita en el propio lenguaje, conteniéndose cada descripción en un fichero distinto. Encontramos muy interesante la idea de separar ambos aspectos ya que, dentro de una UI, una misma funcionalidad puede ser activada desde varios caminos. Separando ambos aspectos evitamos tener que especificar de forma redundante la misma funcionalidad.

4.1. Descripción de la funcionalidad

La funcionalidad que ofrece un servicio se encuentra distribuida entre los dispositivos electrónicos que forman parte del hogar. Desde el punto de vista de los dispositivos, la funcionalidad permite activar el funcionamiento de una característica del dispositivo o leer información relacionada con su estado. Dado que los dispositivos asociados con un mismo servicio comparten la misma funcionalidad, se hace posible en este punto el describir la funcionalidad considerando únicamente el servicio sin tener en cuenta el dispositivo electrónico.

Para describir la funcionalidad que se encuentra dentro de un servicio del hogar se utilizan una serie de elementos que expresan las acciones y respuestas asociadas. A continuación se listan los diferentes elementos que aporta el lenguaje para describir la funcionalidad:

Tabla 1. Elementos del lenguaje para describir la funcionalidad

Acciones	Respuestas
• action • selectAction • optionAction • numericInputAction	• query • simpleResponse • responseRange • response

Las acciones modifican el estado de la funcionalidad implicada. Cada acción tiene asociado un nombre, la información necesaria para activar dicha funcionalidad y un mensaje. Es muy importante que el nombre de la acción sea lo más descriptivo posible, ya que este nombre trascenderá hacia la UI durante la generación automática y, por tanto, será el que acabe llegando al usuario de la aplicación. En cuanto al mensaje, este va dirigido completamente al usuario y debe ofrecer información sobre la acción que se acaba de realizar. Para especificar la funcionalidad asociada a una acción se emplea una pareja de atributos que contienen el nombre de la misma y el valor con el que se activa. La definición de acciones puede ser vista como una forma de dotar de información semántica a funcionalidades que no la tienen, de tal manera que un usuario inexperto pueda hacer uso de ella.

Al contrario que las acciones, las respuestas conservan el estado de la funcionalidad implicada. Las respuestas son empleadas para obtener información sobre el estado en que se encuentran los elementos presentes dentro del hogar. Aportan información directamente al usuario por lo que durante su descripción debe tenerse muy presente a éste último.

Los elementos que componen las acciones y las respuestas se engloban dentro de lo que denominamos servicio. Un servicio puede ser visto, desde el punto de vista del lenguaje, como un espacio en el que la funcionalidad real adopta un significado semántico diferente dependiendo del servicio en que se encuentre. Por ejemplo, dos funcionalidades que compartan el mismo nombre pueden ser descritas de forma diferente en servicios distintos. Desde el punto de vista de la generación automática de la UI, un servicio puede ser visto como una agrupación de acciones y respuestas dentro de un contexto: servicio de iluminación, servicio para el control de puertas, servicio para el control de la calefacción, etc.; las cuales serán accesibles desde la UI.

4.2. Acciones

Como puede observarse en la Tabla 1, existen varios tipos de acciones. El tipo de acción más simple es *action*. Éste aporta información semántica sobre la funcionalidad subyacente, permitiendo activar dicha funcionalidad sin depender del estado actual en que se encuentre. El resto de acciones, más complejas, consisten en la agrupación de elementos *action* con algún tipo de refinamiento.

Al contrario que *action*, el elemento *selectAction* depende completamente del estado actual de la funcionalidad. Debido a esto, este elemento resulta muy útil cuando la activación de la funcionalidad subyacente solo tiene sentido según el estado actual. Por ejemplo, si una luz está encendida, que la interfaz de usuario muestre una acción para encender la luz no tiene ningún sentido bajo esas condiciones, siendo la acción más natural la de apagar la luz. El elemento *selectAction* está compuesto por dos o más elementos *action*. Sin embargo, durante la instanciación de la interfaz, solo uno de los dos elementos *action* será accesible por la interfaz dependiendo del estado en que se encuentre la funcionalidad indicada.

Hay ocasiones en que la funcionalidad subyacente puede activarse empleando un amplio rango de valores. Supongamos que la funcionalidad de una persiana permite establecer su posición numéricamente empleando un valor comprendido entre 0 y 255, que determine, respectivamente, si la persiana está bajada o subida completamente. En un caso como este, donde la persiana puede situarse en cualquiera de esas posiciones, es posible reducir el número de posiciones a sólo unas cuantas que resulten útiles en la práctica: bajada, subida o a media altura. Esto es justo lo que se pretende conseguir con el elemento *optionAction*. Dada una funcionalidad con soporte de un amplio rango de valores, con este elemento se pueden especificar los valores que son más utilizados en la práctica, agrupando las opciones que actúan sobre la misma funcionalidad para no perder la relación existente entre los elementos.

Sin embargo, también hay ocasiones en que es necesario poder activar una funcionalidad empleando cualquier valor dentro de su rango permitido, como por ejemplo, en el caso de un regulador de temperatura. El elemento *numericInputAction* permite activar la funcionalidad con cualquier valor del rango mediante incrementos y decrementos del valor actual de la misma. Este elemento está compuesto por dos elementos *action*, uno para incrementar el valor actual y otro para decrementarlo. La cantidad en que se incrementa o decrementa el valor actual se especifica en la descripción del elemento. No obstante, puede darse el caso de que a la funcionalidad subyacente no se le pueda indicar el incremento de forma relativa al valor actual, es decir, indicar como valor de activación un valor numérico que se sume al valor actual de la funcionalidad. En esos casos, es necesario que la propia UI recupere el valor actual de la funcionalidad para luego calcular el valor absoluto tras el incremento y, posteriormente, emplearlo como valor de activación. Previendo la falta de soporte por parte de la funcionalidad subyacente del dispositivo, el elemento *numericInputAction* recoge en su descripción esta información para determinar si es necesario calcular el valor absoluto antes de activar la funcionalidad.

Cabe destacar que el modo de funcionamiento del elemento *numericInputAction* es reactivo, es decir, tan pronto el usuario realiza cambios en el valor de la funcionalidad, ésta se activa inmediatamente. Para retrasar la activación real de la funcionalidad, evitando que los cambios en el valor se apliquen inmediatamente, se puede alterar el modo de funcionamiento para que una vez seleccionado el valor adecuado, la UI solicite confirmación al usuario.

4.3. Respuestas

Aunque todas las acciones revisadas hasta el momento ofrecen una respuesta al usuario aportando información sobre el estado de la acción realizada, hasta ahora no se ha presentado ningún mecanismo que de forma explícita permita recuperar información real sobre los elementos presentes en el hogar. El elemento *query* aporta al lenguaje este mecanismo. A través de este elemento es posible realizar consultas sobre el estado de luces, persianas, puertas, sensores de temperatura, etc., presentes en el hogar.

El elemento *query* contiene el nombre de la consulta y la funcionalidad que necesita para realizar la misma. El nombre debe describir la consulta que realiza de la mejor manera posible ya que, al igual que ocurre en el elemento *action*, éste formará parte de la UI tras la generación automática. La funcionalidad empleada para realizar la consulta devuelve como respuesta un valor numérico que debe ser interpretado. Por este motivo, todo elemento *query* lleva asociado un elemento respuesta que contiene la información necesaria para interpretar correctamente el valor devuelto como respuesta. La interpretación consiste en mapear o relacionar los valores numéricos obtenidos como respuesta con cadenas de texto. Por tanto, estas cadenas de texto deben ser muy descriptivas pues será ésta la información que acabe recibiendo el usuario.

La respuesta más simple viene dada por el elemento *simpleResponse*. Este elemento es un contenedor de respuestas que relacionan valores

numéricos con cadenas de texto. Cuando la misma respuesta puede darse dentro de un dominio se emplea el elemento *responseRange*. Este elemento permite relacionar rangos numéricos no solapados con cadenas de texto. Algunos tipos de consulta emplean funcionalidades que devuelven como resultado una respuesta numérica que no necesita interpretación, como por ejemplo, la consulta de la temperatura. En esos casos es necesario encapsular la respuesta dentro de una cadena de texto para que el mensaje sea más amigable para el usuario. El elemento *response* permite encapsular la respuesta obtenida dentro de un mensaje. Para ello, utiliza dentro del mensaje una variable que recibe su valor real en el momento de ejecutarse la consulta.

4.4. Descripción de la interfaz de usuario

Para describir la interfaz de usuario se emplea lo que denominamos listas de navegación. Una lista de navegación es un contenedor de elementos navegables a través de los cuales se puede acceder a otras listas de navegación. Es posible visualizar en forma de árbol la estructura que se forma al emplear estas listas. Entre los elementos empleados para describir la interfaz de usuario hacemos una distinción entre aquellos que en tiempo de ejecución no modifican la estructura del árbol y aquellos que sí la modifican. Los primeros reciben el nombre de elementos estáticos, mientras que a los segundos nos referimos como elementos dinámicos.

Tabla 2. Elementos del lenguaje para describir la Interfaz de Usuario

Elementos Estáticos	Elementos Dinámicos
• navigation • menu • launcher	• loadLaunchers

4.5. Elementos estáticos

Dentro de los elementos estáticos se encuentra el elemento *navigation* para describir listas de navegación. Cada lista de navegación tiene asociada una descripción que aporta información al usuario sobre el contenido de la misma. Existen variables predefinidas que pueden ser empleadas en la descripción de la lista para indicar, por ejemplo, el número de elementos que ésta contiene. Las listas de navegación son identificadas de forma unívoca por medio de un identificador, el cual es empleado por los elementos navegables como se verá más adelante.

Un elemento navegable tiene asociado un nombre. Este nombre es empleado en la generación de la UI y, por tanto, es muy importante que transmita de forma concisa la información que puede encontrarse si se navega por el mismo. Un elemento navegable puede ser considerado de dos formas distintas según sea un nodo intermedio del árbol o un nodo hoja. En el primer caso, el elemento navegable es considerado un menú (elemento *menu*). Mientras que en el segundo caso, el elemento navegable es considerado un lanzador (elemento *launcher*).

Como menú, un elemento navegable actúa como punto de acceso hacia otra lista de navegación. Así, un menú se asocia con una lista de navegación por medio del identificador de la misma. Un menú, de forma opcional, puede contener argumentos, cadenas de texto que son transmitidas a la lista de navegación asociada cuando el usuario entra en ella. Estos argumentos pueden ser empleados por la lista de navegación en su descripción o en los elementos navegables que contiene. El poder comunicar cierta información entre listas de navegación hace posible la descripción de UIs que no sería posible de otra manera. Por ejemplo, con esta comunicación entre listas, es posible crear listas de navegación genéricas que, dependiendo de los argumentos recibidos, describan UI distintas.

Cuando el elemento navegable es considerado un lanzador, éste actúa como punto de entrada a uno de los servicios definidos previamente. Sin embargo, falta especificar una relación muy importante para que el lanzador realice su función correctamente. Como ya se ha comentado, la funcionalidad descrita en los servicios realmente reside en los dispositivos electrónicos que se encuentran dentro del hogar. Por tanto, se hace necesario vincular el lanzador con el dispositivo real. Utilizando argumentos predefinidos en los lanzadores se realiza la asociación con los dispositivos y, cuando el usuario ejecuta el lanzador, esta información es transmitida al servicio. De esta manera, la funcionalidad del servicio que fue definida independientemente del dispositivo, contiene en tiempo de ejecución toda la información necesaria para ser activada. Puesto que hay dispositivos que a pesar de pertenecer al mismo servicio no soportan toda la funcionalidad disponible, los lanzadores pueden contener esta información también en sus argumentos para que a través del servicio esas funcionalidades no estén presentes.

4.6. Elementos dinámicos

Con el objetivo de hacer más potente y flexible la forma en que se describe la interfaz de usuario, se ha incorporado al lenguaje un mecanismo que permite añadir lanzadores a las listas de navegación en tiempo de ejecución. Este mecanismo es el elemento *loadLaunchers*. A través del mismo se pueden agregar lanzadores, previamente definidos, que satisfagan correctamente un criterio de búsqueda.

Los lanzadores son definidos a parte, en otro fichero, junto a etiquetas que los clasifican. Para

añadirlos a una lista de navegación, empleando el elemento *loadLaunchers*, se especifica el fichero donde se encuentran definidos y el filtro o condición de búsqueda que debe satisfacerse.

Manteniendo la definición de los lanzadores separada de la definición de la UI, se permite la reutilización de los mismos en diferentes UIs. Además, este mecanismo hace factible la creación de procesos que mantengan una base de datos actualizada de los lanzadores disponibles. De esta manera, una UI se pueda mantener actualizada sin realizar cambios en la descripción de la misma.

5. Caso de estudio: interfaz para personas ciegas

Como parte del presente trabajo se ha realizado un caso de estudio en el que se ha desarrollado una interfaz móvil para personas con discapacidad visual. A través de esta interfaz se permite el acceso remoto a toda la funcionalidad y servicios ofrecidos por el hogar digital, aumentando, de este modo, la autonomía de las personas dentro de sus hogares.

El desarrollo de esta interfaz se ha realizado empleando el lenguaje de descripción de interfaces de usuario introducido en la sección 4. Por tanto, un paso previo al desarrollo de la interfaz ha sido la implementación del intérprete que la genera. Cabe destacar que una vez se cuenta con una implementación funcional del intérprete, el desarrollo de UIs accesibles para el hogar inteligente se reduce a describir el modo de navegación y la funcionalidad de los servicios a los que acceder.

5.1. Detalles del intérprete

El intérprete ha sido desarrollado para la plataforma móvil Android de la Open Handset Alliance (OHA), liderada por Google. Esta plataforma ha sido escogida frente a otras por diversos motivos. Por un lado, su filosofía abierta hace que prácticamente no tenga barreras de entrada para comenzar a desarrollar. Mientras que por otro, su fuerte crecimiento y rápida expansión, nos asegura la existencia de una gran base de usuarios procurando que las aplicaciones desarrolladas para esta plataforma lleguen a un gran número de ellos.

Debido a que Android funciona, principalmente, en smartphones, el intérprete se ha diseñado teniendo en cuenta las características más comunes que se encuentran en estos dispositivos. Por tanto, se hace uso de la conectividad inalámbrica y la pantalla táctil. La pantalla es utilizada, en lugar de para la visualización de la UI, para la interacción del usuario con la aplicación. A tal efecto, se ha implementado el soporte para cuatro gestos (ver Tabla 3) que el usuario puede emplear para desplazarse entre las listas de navegación y los elementos navegables. Los gestos han sido diseñados para que sean fáciles de recordar y de realizar sin tener referencias visuales.

Cada uno de los gestos se puede realizar manteniendo el dedo pulgar sobre la pantalla, deslizándolo en cualquiera de las cuatro direcciones que aparecen en la Tabla 3. El gesto puede iniciarse desde cualquier punto de la pantalla, por lo que no es necesario que el usuario aprenda una posición inicial a partir de la cual iniciar el gesto.

Tabla 3. Gestos de navegación

Gesto	*Acción asociada*
Derecha	Siguiente elemento
Izquierda	Elemento anterior
Arriba	Volver a la lista anterior
Abajo	Entrar / Ejecutar

La interacción entre la aplicación y el usuario se realiza convirtiendo los mensajes textuales en voz mediante alguno de los sintetizadores de voz disponibles. En este caso de estudio se han implementado tres modos diferentes de sintetizar voz. El primero de ellos utiliza el sintetizador de voz ofrecido por la plataforma Android. Sin embargo, en algunos terminales móviles la conversión de texto en voz no se realiza lo suficientemente rápido como para emplearse en una aplicación de este tipo. Por este motivo, se ha realizado una segunda implementación que accede a un sintetizador de voz externo a través de la conexión inalámbrica. Finalmente, la última implementación consiste en ofrecer como servicio del hogar la conversión de texto a voz. De esta manera, los mensajes auditivos de la aplicación se reproducen por el sistema de sonido del hogar inteligente.

Por tanto, utilizando los mecanismos descritos de interacción, el intérprete ofrece al usuario el soporte de accesibilidad necesario para manejar la UI mediante gestos y recibir información auditiva sobre las acciones que realiza mediante voz.

El intérprete implementado emplea como entrada la descripción de la UI y la descripción de la funcionalidad disponible en los servicios del hogar inteligente. A partir de estas descripciones es capaz de generar de forma automática la interfaz que utilizará el usuario. La UI que produce internamente el intérprete se basa en una representación textual, pensada para que sea accesible utilizando cualquiera de las tres implementaciones de sintetizadores de voz. Una vez construida la UI, el intérprete reacciona ante las acciones del usuario de dos modos distintos: permitiendo que el usuario se desplace a través de las listas de navegación y elementos navegables; o realizando acciones directamente sobre los dispositivos del hogar.

En el primer caso, cada vez que el usuario visita un elemento o pasa de una lista de navegación a otra,

el intérprete reproduce una locución transmitiendo al usuario el mensaje correspondiente a la navegación que está realizando. Además, puesto que el intérprete es consciente del número de elementos navegables presentes en cada lista de navegación, cuando el usuario alcanza el último elemento navegable y pretende ir más allá, se le avisa mediante un mensaje de voz predefinido, volviendo al primer elemento navegable. Lo mismo ocurre en el sentido contrario, cuando el usuario alcanza el primer elemento navegable y pretende visitar el anterior.

En el segundo caso, cuando el usuario activa una funcionalidad desde la UI, el intérprete ejecuta esa acción en el dispositivo real. Tras la ejecución, el usuario recibe el mensaje de voz asociado a la acción que acaba de realizar y que está previamente definido en la descripción de la funcionalidad.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<userinterface start="main" version="0.1">
    <navigation
        id="main"
        desc="Welcome to DAI Home.
              There are ${count} available options">
        <menu
            name="1) Home Devices"
            goto="devices"
            argc="1"
            argv="home devices">
        </menu>
        <menu
            name="2) Scenes"
            goto="scenes">
        </menu>
        <menu
            name="3) Application Settings"
            goto="settings">
        </menu>
    </navigation>
    <navigation
        id="devices"
        desc="There are ${count} available ${1}">
        <loadLaunchers file="launchers.xml" filter="none" />
    </navigation>
    // Other navigation lists
</userinterface>
```

Imagen 1. Descripción de la UI

5.2. Descripción de la Interfaz

La interfaz para personas con discapacidad visual se ha desarrollado utilizando el UIDL presentado en este trabajo. La descripción realizada define una UI que puede controlar toda la funcionalidad presente en los servicios del hogar inteligente. En ese sentido, algunas de las guías para diseñar UIs para personas ciegas presentadas en [21] han sido tenidas en cuenta. La descripción de la UI contiene cuatro listas de navegación. En la Imagen 1 se encuentra, a modo de ejemplo, un extracto donde se muestra la descripción de la primera. Esta lista es considerada la principal y es la que el intérprete presenta inicialmente al usuario mediante un mensaje de voz que describe su contenido. La lista principal contiene tres elementos navegables: dispositivos, escenas y configuración. A través del primer elemento se alcanza una lista de navegación que contiene lanzadores para todos los dispositivos presentes en el hogar. Mediante el segundo elemento se alcanza otra lista con lanzadores para cada una de los escenarios configurados en el hogar inteligente. Finalmente, utilizando el tercer elemento se alcanza una lista con lanzadores para configurar aspectos de la propia aplicación.

Cuando el usuario visita un lanzador, el intérprete presenta a éste las acciones que pueden realizarse desde el mismo. Las acciones que puede realizar el usuario vienen recogidas en la descripción de la funcionalidad del servicio (ver Imagen 2). En la descripción de esta funcionalidad se ha empleado un vocabulario breve y conciso en los nombres, normalmente empleando un verbo y un sustantivo. Así, por ejemplo, para describir la funcionalidad que enciende la luz dentro del servicio de iluminación, se ha empleado como nombre “encender luz”.

Un aspecto que se ha tenido en cuenta en el intérprete es el siguiente. Puesto que un lanzador contiene información sobre la funcionalidad que puede emplear, el intérprete no debe presentar al usuario aquellas acciones cuya funcionalidad no está presente en el lanzador. De esta manera, aunque el servicio sea capaz de realizar una gran cantidad de acciones, a través del lanzador únicamente se presentarán aquellas que realmente éste último es capaz de realizar.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<services version="0.1">
  <service
    id="Lighting"
    name="Lighting Service">
    <selectAction func="dataSwitch">
        <action
            name="Turn light on"
            func="switch"
            value="1"
            compare="0"
            onSuccess="The light is on" />
        <action
            name="Turn light off"
            func="switch"
            value="0"
            compare="1"
            onSuccess="The light is off" />
    </selectAction>
    <query
        name="Inquire light state"
        func="dataSwitch">
        <simpleResponse>
            <response value="1" message="The light is on" />
            <response value="0" message="The light is off" />
        </simpleResponse>
    </query>
    // Other  services
  </service>
</services>
```

Imagen 2. Descripción de funcionalidad

En la Imagen 3 se muestra una descripción para tres lanzadores que se corresponden con dispositivos del hogar. Como se observa, cada lanzador está asociado a un servicio y tiene especificada la funcionalidad que necesita para realizar su función.

El papel de los lanzadores dentro del UIDL es fundamental. La función del lanzador es la de vincular la descripción de la UI con la descripción de la funcionalidad de los servicios del hogar inteligente, facilitando así la reutilización de descripciones por otras UI.

Debido al lenguaje UIDL y el soporte ofrecido por el intérprete, a partir de la descripción de la interfaz para personas con discapacidades visuales se genera automáticamente una interfaz accesible para acceder de forma remota a los servicios del hogar inteligente.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<launchers version="0.1">
   <launcher
      id="1"
      name="Bedside Light"
      service="Lighting"
      tag="Sittingroom"
      func="switch,dataSwitch,toggle">
   </launcher>
   <launcher
      id="8"
      name="Blind"
      service="Blinds"
      tag="Kitchen"
      func="moveBlind,dataPosBlind,posBlind">
   </launcher>
   <launcher
      id="11"
      name="Door"
      service="Doors"
      tag="Bathroom"
      func="switch,dataSwitch,toggle">
   </launcher>
</launchers>
```

Imagen 3. Descripción de lanzadores

6. Prototipo

Este trabajo ha sido incorporado al proyecto metalTIC – Hogar digital (http://www.metaltic.org), realizado por el grupo de Domótica y Ambientes Inteligentes (DAI) de la Universidad de Alicante y la Federación de Empresarios del Metal de la Provincia de Alicante (FEMPA). Este es un laboratorio demostrativo que simula una vivienda con los últimos avances en las tecnologías de la información, las comunicaciones y el control para aportar servicios en el hogar (ver Imagen 4).

La casa de unos 50 metros cuadrados cuenta con un salón con cocina integrada, un dormitorio y un cuarto de baño donde se han integrado las tecnologías más recientes e innovadoras que aportan servicios a la vivienda, articulando conceptos clave como el control energético, la vigilancia y la seguridad, el confort y el ocio, las comunicaciones, la accesibilidad y la administración.Incorpora las últimas tecnologías tanto en interacción persona-entorno (televisión, pantalla táctil, dispositivos de visión, reconocimiento de voz, dispositivos móviles, acelerómetros, giroscopios,…) como en sistemas de control domótico para la realización de investigación tanto básica como aplicada.

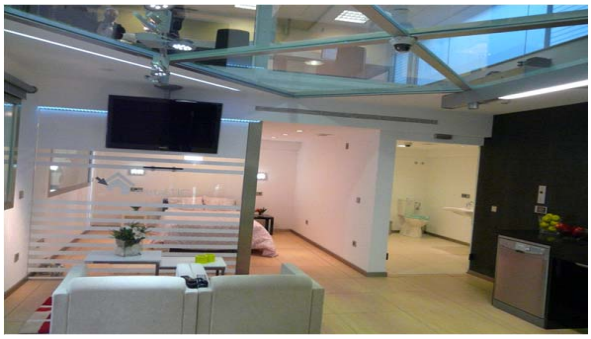

Imagen 4. metalTIC – Hogar Digital

Integra las tecnologías abiertas KNX, LonWorks y DALI (iluminación), las tecnologías inalámbricas Zigbee y EnOcean, y las tecnologías propietarias INELI (empresa asociada a FEMPA) y Fagor (electrodomésticos). Para su integración se emplean coordinadamente un autómata programable con las prestaciones de un sistema informático basado en una arquitectura PC embebida y un sistema Windows Media Center. Incorpora el middleware destinado a independizar los aspectos más técnicos de los niveles más altos de aplicación asegurando la interoperabilidad entre todos los sistemas de control indicados anteriormente. Asimismo, este middleware oferta toda la funcionalidad de metalTIC mediante servicios web, para permitir que de una forma sencilla se desarrollen nuevos servicios e interfaces persona-entorno y se validen posteriormente. En conclusión, este espacio ofrece importantes facilidades para realizar en ella nuevos desarrollos y pruebas de usuario.

7. Conclusiones y trabajos futuros

En este trabajo se ha presentado un lenguaje para describir UI que son empleadas en aplicaciones de la vida asistida por el entorno. El lenguaje ofrece mecanismos diferentes para describir, por un lado, la funcionalidad de los servicios del hogar digital, y por otro, la UI que accede a esos servicios; estando ambas descripciones conectadas mediante lo que se ha denominado lanzador. En el caso de estudio presentado, este lenguaje ha mostrado ser suficiente para desarrollar rápidamente una interfaz para personas con discapacidad visual. A través de esta interfaz se puede acceder a los servicios del hogar inteligente utilizando un smartphone. Sin embargo, la validación por usuarios reales en metalTIC – Hogar digital, es una tarea pendiente.

Un aspecto sobre el que puede profundizarse para aumentar la usabilidad de las UI es el estudio de modos de locución automática en el intérprete para leer automáticamente todos los elementos de

navegación de cada lista. Estudiar el impacto de estos modos sobre la usabilidad es un aspecto a tener en cuenta. En cuanto al lenguaje, estudiar nuevas abstracciones independientes de la modalidad se hace necesario para enriquecer y aumentar su capacidad para describir UIs. Con la especificación actual, el UIDL presentado permite describir UIs empleando la metáfora de las listas. Sin embargo, se podrían incorporar de forma alternativa recursos gráficos, como pictogramas, abriendo la puerta de esta manera a nuevas representaciones gráficas. Finalmente, un trabajo interesante sería el portar a otras plataformas el intérprete implementado en este trabajo, o igualmente, crear otros intérpretes para otras modalidades.

8. Agradecimientos

Se agradece la colaboración de la Federación de Empresarios del Metal de la Provincia de Alicante (FEMPA) por la cesión de metalTIC – Hogar Digital para la validación de este trabajo.

10. Referencias

[1] B.R. Connell, M. Jones, R. Mace, J. Mueller, A. Mullick, E. Ostroff, J. Sanford, E. Steinfeld, M. Story y G. Vanderheiden, “The Principles Of Universal Design”, The Center for Universal Design, NC State University, USA, 1997, pp. 2-4.

[2] A. Hardwick, S. Furner y J. Rush, “Tactile display of virtual reality from the World Wide Web – a potential access method for blind people”, Displays, vol. 18, 1998, p. 153–161.

[3] A. Savidis y C. Stephanidis, “The HOMER UIMS for dual user interface development: Fusing visual and non-visual interactions”, Interacting with Computers, vol. 11, Dec. 1998, pp. 173-209.

[4] N. Souchon y J. Vanderdonckt, “A review of xml-compliant user interface description languages”, Interactive Systems. Design, Specification, and Verification, 2003, p. 391–401.

[5] J. Guerrero-Garcia, J.M. Gonzalez-Calleros, J. Vanderdonckt y J. Munoz-Arteaga, *A Theoretical Survey of User Interface Description Languages: Preliminary Results*, IEEE, 2009.

[6] G.E. Pfaff, *User Interface Management Systems*, Berlín: Springer-Verlag, 1985.

[7] G. Calvary, “A Unifying Reference Framework for multi-target user interfaces”, Interacting with Computers, vol. 15, 2003, pp. 289-308.

[8] M. Abrams, “UIML: an appliance-independent XML user interface language”, Computer Networks, vol. 31, 1999, pp. 1695-1708.

[9] K. Coninx, K. Luyten y K.A. Schneider, eds., “Task Models and Diagrams for Users Interface Design”, 5th International Workshop, TAMODIA 2006, Hasselt, Belgium, Revised Papers, TAMODIA, Springer, 2007.

[10] J. Eisenstein y J. Vanderdonckt, “Adapting to mobile contexts with user-interface modeling”, 2000, pp. 83-92.

[11] A. Puerta y J. Eisenstein, *XIML*, New York, New York, USA: ACM Press, 2002.

[12] Q. Limbourg, J. Vanderdonckt y B. Michotte, “UsiXML: A user interface description language for context-sensitive user interfaces”, Interface Description, 2004, p. 55–62.

[13] A. Arsanjani, D. Chamberlain, D. Gisolfi, R. Konuru, J. Macnaught, S. Maes, R. Merrick, D. Mundel, T. Raman, S. Ramaswamy, T. Schaeck, R. Thompson, A. Diaz, J. Lucassen y C.F. Wiecha, (WSXL) Web Service Experience Language Version 2, IBM, 2002.

[14] A.S. Incorporated, “Developing applications in MXML”, Using Flex 4, 2010, pp. 10-27.

[15] Microsoft, “XAML Overview (WPF)” MSDN. disponible en http://msdn.microsoft.com/en-us/library/ms752059.aspx (acceso 02/04/2011)

[16] N. Deakin, “XUL Tutorial”, Mozilla. disponible en https://developer.mozilla.org/en/XUL_Tutorial (acceso 02/04/2011)

[17] Nokia, “Introduction to the QML Language” 2010. disponible en http://doc.qt.nokia.com/4.7-snapshot/qdeclarativeintroduction.html (acceso 02/04/2011)

[18] G. Zimmermann, G. Vanderheiden y A. Gilman, “Universal remote console - prototyping for the alternate interface access standard”, Universal Access Theoretical Perspectives, Practice, and Experience, 2003, p. 524–531.

[19] J. Nichols, B. a Myers, M. Higgins, J. Hughes, T.K. Harris, R. Rosenfeld y M. Pignol, “Generating remote control interfaces for complex appliances”, Proceedings of the 15th annual ACM symposium on User interface software and technology - UIST ’02, 2002, p. 161.

[20] J. Nichols, B.A. Myers, K. Litwack, M. Higgins, J. Hughes y T.K. Harris, “Describing appliance user interfaces abstractly with xml”, Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages, Citeseer, 2004, p. 9–16.

[21] S. Leuthold, J. Bargasavila y K. Opwis, “Beyond web content accessibility guidelines: Design of enhanced text user interfaces for blind internet users”, International Journal of Human-Computer Studies, vol. 66, Apr. 2008, pp. 257-270.

[22] B. Lucas, “VoiceXML for Web-based distributed conversational applications”, Communications of the ACM, vol. 43, Sep. 2000, pp. 53-57.