Diseño del sistema

Tabla de contenido

[1. Introducción 3](#_Toc406053328)

[2. Repaso de la arquitectura general 3](#_Toc406053329)

[2.1. Capa de modelado (Modelling Layer) 5](#_Toc406053330)

[2.1.1. El Capabilities Collector 5](#_Toc406053331)

[2.1.2. El Semantic Modeller 5](#_Toc406053332)

[2.2. Capa de adaptación (Adaptation Layer) 6](#_Toc406053333)

[2.2.1. El Adaptation Engine 6](#_Toc406053334)

[2.2.2. El Adaptation Polisher 6](#_Toc406053335)

[2.2.3. Capa de aplicación (Application Layer) 8](#_Toc406053336)

[3. Diseño del módulo de caracterización de usuarios, dispositivos y contexto 9](#_Toc406053337)

[3.1. Usuario 13](#_Toc406053338)

[3.2. Contexto 16](#_Toc406053339)

[3.3. Dispositivo 24](#_Toc406053340)

[4. Diseño del módulo de aprendizaje automático 26](#_Toc406053341)

[1.1. Capa de transformación 26](#_Toc406053342)

[1.2. Capa de aprendizaje 27](#_Toc406053343)

[1.2.1. Descubrimiento de los conjuntos de acciones frecuentes 28](#_Toc406053344)

[1.2.2. Descubrimiento de la topología 29](#_Toc406053345)

[1.2.3. Descubrimiento de las relaciones temporales 29](#_Toc406053346)

[1.2.4. Descubrimiento de las condiciones 30](#_Toc406053347)

[1.3. Capa de aplicación 30](#_Toc406053348)

[4.1. Diseño del módulo de detección de cambios comportamentales 30](#_Toc406053349)

[a. Calculando probabilidades 31](#_Toc406053350)

[b. Calculo de posible modificaciones 31](#_Toc406053351)

[5. Diseño del módulo de adaptación del conocimiento 32](#_Toc406053352)

[6. Diseño del módulo para la generación automática de funciones de pertenencia 33](#_Toc406053353)

[7. Diseño del módulo de procesado y adaptación 34](#_Toc406053354)

[8. Diseño del módulo de análisis del comportamiento 36](#_Toc406053355)

[9. Implementación del Módulo de aprendizaje automático 37](#_Toc406053356)

[10. Referencias 38](#_Toc406053357)

## Introducción

El presente documento contiene el diseño de los diferentes módulos que componen el sistema. Estos módulos fueron implementados en la Fase 3 del proyecto, sirviendo el presente documento como guía en el proceso. Además, tal y como se especificó en la Fase 2 del proyecto, se han realizado una serie de **actualizaciones** sobre el estado de este documento, aportando las características actualizadas del diseño final del sistema.

## Repaso de la arquitectura general

Como ya se indicó en el entregable “Elección de tecnologías”, la adaptación que se debe de llevar a cabo depende de cuatro factores:

* *El contexto actual en el que se está usando la aplicación*. Por ejemplo en caso de que el usuario se encuentre en algún lugar con mucho ruido ambiental (una discoteca, un concierto…) las aplicaciones deberán mostrar una interfaz basada en la visión.
* *Las características del usuario que está utilizando la aplicación*. Si el usuario que utiliza la aplicación sufre algún tipo de discapacidad (por ejemplo si el usuario es ciego), la interfaz de la aplicación se deberá adaptar para poder ofrecerle el mejor servicio posible.
* *Las preferencias del usuario que está utilizando la aplicación*. El usuario puede seleccionar entre diferentes opciones. A partir de esa selección, el contexto y las características del usuario se crea un perfil personalizado por cada usuario que determina sus preferencias. Como las preferencias del usuario, así como el contexto pueden cambiar a lo largo del tiempo, la respuesta del usuario a la adaptación se utiliza para modificar el perfil del usuario y proveer soluciones acordes a las preferencias cambiantes.
* *Las características del dispositivo donde se ejecuta*. El ecosistema actual de dispositivos es muy variado, con múltiples tamaños de pantalla, resoluciones, codificadores multimedia, velocidades de procesador, tamaños de memoria… Por lo tanto, la interfaz de la aplicación deberá de modificar su comportamiento para adaptarse al dispositivo en el que se está ejecutando.

Combinando estos cuatro factores, el módulo de adaptación dinámica infiere cuál es la adaptación que debe llevarse a cabo para poder ofrecer el servicio de la mejor manera, ofreciéndole al usuario la mejor experiencia posible.



Figura 1. Proceso de adaptación de la interfaz del usuario según la situación del usuario, contexto y características del dispositivo actual.

El diseño de la arquitectura de DYNUI se resume en el siguiente diagrama:

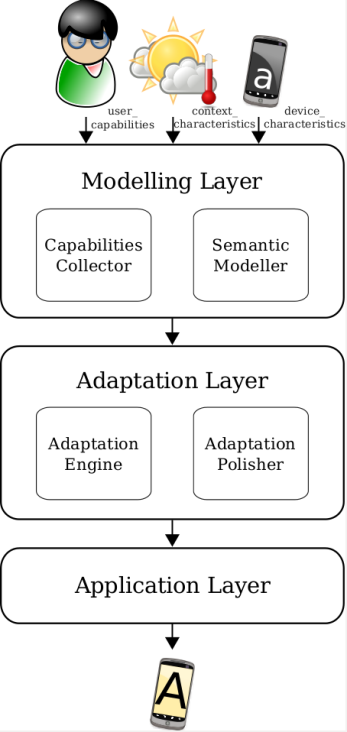


Figura 2. Arquitectura de DYNUI.

A continuación se detallan tanto las capas que forman el diseño final de la arquitectura de DYNUI como los módulos que la componen.

## Capa de modelado (Modelling Layer)

La capa de modelado tiene como objetivo la generación de los diferentes modelos semánticos que conforman las entradas al sistema DYNUI. Estos modelos se corresponden con las entidades *usuario*, *contexto* y *dispositivo*. Para conseguir un modelo consistente, la capa de modelado combina el resultado de dos módulos. Por un lado, el denominado *Capabilities Collector*, y por otro el *Semantic Modeller*.

## El Capabilities Collector

Este módulo se encarga principalmente de capturar aquellas capacidades de usuario que se vean afectadas por el contexto y el dispositivo actuales. De este modo, el *Capabilities Collector* es capaz de construir un modelo interno (no semántico) en el dispositivo. Este modelo es suficientemente ligero para ser tratado en un tiempo muy asequible, lo cual aporta ligereza al resto del sistema.

## El Semantic Modeller

Una vez el *Capabilities Collector* ha terminado de construir el modelo, éste es enviado al *Semantic Modeller*. Este módulo es capaz de traducir el modelo generado en un modelo semántico, el cual se almacenará también en el dispositivo. Al no existir herramientas semánticas suficientemente fiables para dispositivos móviles, DYNUI incorpora *Pellet4Android*, un motor de razonamiento semántico basado en Pellet totalmente compatible con dispositivos Android. *Pellet4Android* es un proyecto open source, disponible para su descarga en el siguiente enlace: <https://github.com/edlectrico/Pellet4Android>

## Capa de adaptación (Adaptation Layer)

Esta capa es la encargada de realizar las adaptaciones pertinentes en la interfaz de usuario. Formada por el motor de adaptación (*Adaptation Engine*) y el refinador de adaptación (*Adaptation Polisher*) su objetivo es la adaptación dinámica, en tiempo de ejecución, de dicha interfaz.

## El Adaptation Engine

El guardado del conocimiento correspondiente por el Semantic Modeller en la ontología diseñada para AdaptUI provoca el lanzamiento de una serie de reglas. Estas reglas modifican la clase *Adaptation*, la cual almacena los cambios correspondientes para cada tipo de elemento mostrado en la interfaz de usuario. Una vez esta clase ha sido editada con los cambios correspondientes de la ejecución de estas reglas, el *Adaptation Engine* se encarga de realizar las correspondientes *queries* a la ontología. Es entonces cuando el *Adaptation Engine* lanza una serie de métodos cuyo objetivo son la adaptación dinámica de los elementos representados por la clase *Adaptation*.

## El Adaptation Polisher

Una vez el *Adaptation Engine* termina su tarea, se monitoriza la interacción del usuario mediante la evaluación de ciertas métricas de usabilidad. Estas métricas establecen el nivel de interacción que el usuario mantiene con la aplicación actual. La labor del *Adaptation Polisher* es la de alertar al sistema DYNUI de posibles bajones en la interacción. De este modo, el propio *Adaptation Polisher* es capaz de, ejecutando un conjunto de reglas de urgencia, realizar pequeños cambios en la interfaz de usuario hasta volver a elevar los resultados de la evaluación de estas reglas. Este módulo actúa, pues, como salvavidas para posibles casos en los que alguno de los anteriores módulos haya cometido algún error, habiendo conllevado esto a una interfaz de usuario resultante errónea. A continuación se muestran las métricas de usabilidad utilizadas por el *Adaptation Polisher*, tal y como aparecen en el documento ISO/IEC 9126-4:

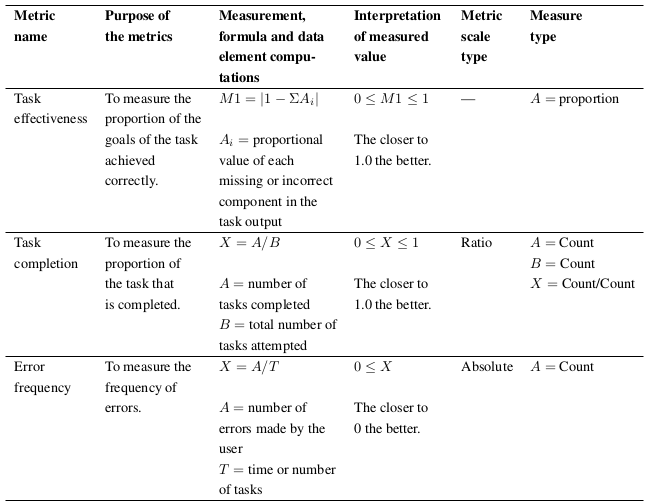


Figura 3. Métricas de usabilidad.

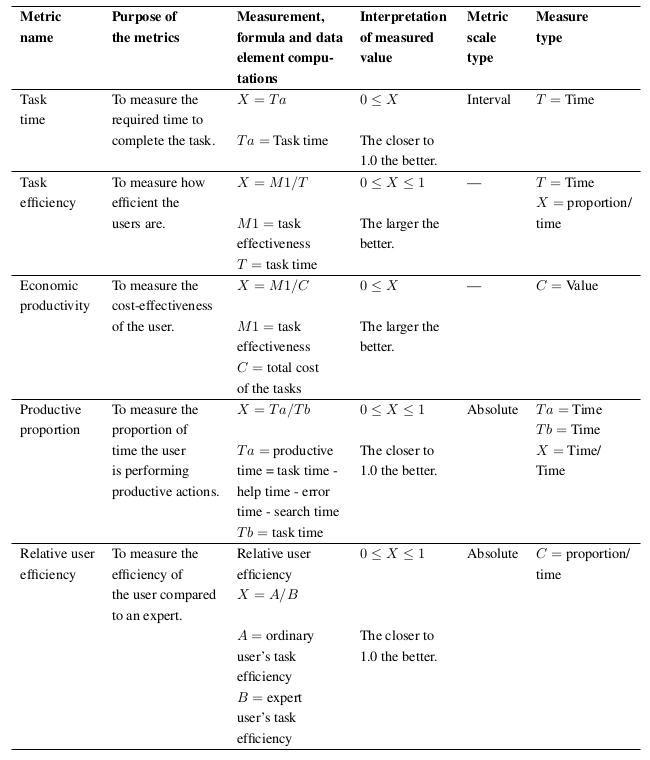


Figura 4. Métricas de productividad.

## Capa de aplicación (Application Layer)

Con el fin de extender y liberar el acceso a frameworks de adaptación dinámica de interfaces de usuario, en la arquitectura de DYNUI se ha incluido una capa de aplicación, la cual provee un API para el desarrollo de interfaces adaptables para dispositivos móviles. Aprovechando las bondades del motor *Pellet4Android*, DYNUI no sólo ofrece un API para manejar la adaptación dinámica de los elementos de la interfaz, sino también una serie de métodos cuyo objetivo no es otro que modificar el conocimiento representado en la ontología. Esto se consigue gracias a la utilización de las librerías correspondientes de OWL-API. Las siguientes tablas listan los métodos más interesantes de ambas API.

Tabla 1. Métodos para la adaptación.

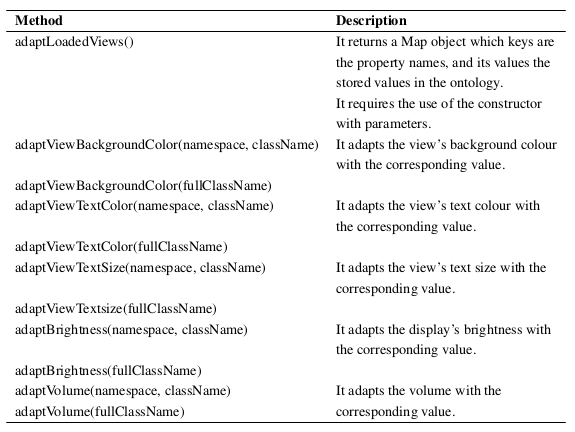
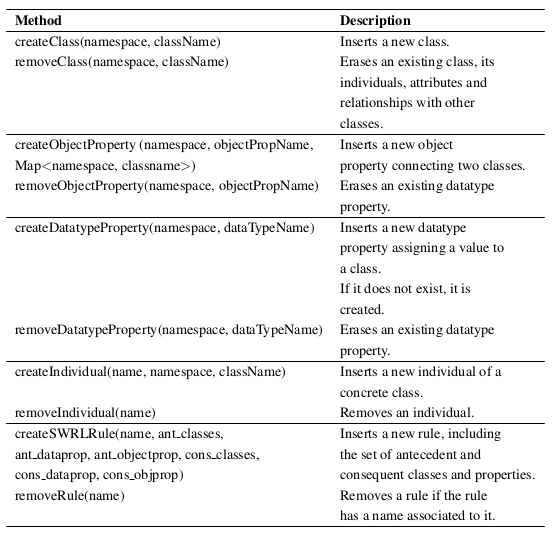


Tabla 2. Métodos para la edición del conocimiento.



## Diseño del módulo de caracterización de usuarios, dispositivos y contexto

Para trabajar con usuarios y sus capacidades se ha elegido como base la “Clasificación Internacional del Funcionamiento y la Discapacidad” (CIDDM), documento generado bajo supervisión de la Organización Mundial de la Salud que clasifica sistemáticamente cualquier estado funcional asociado con estados de salud (ej. enfermedades, trastornos, lesiones, traumas o cualquier otro estado de salud).

Los principales objetivos de la clasificación pueden resumirse de la siguiente manera:

1. Proporcionar una base científica para entender y estudiar los estados funcionales asociados con los estados de salud;
2. Establecer un lenguaje común para describir los estados funcionales asociados con estados de salud, con el fin de mejorar la comunicación entre los profesionales de la salud o en otros sectores, y personas con discapacidad;
3. Permitir la comparación de datos entre países, o entre disciplinas relacionadas con la
4. atención médica, entre los servicios, y en diferentes momentos a lo largo del tiempo;
5. Proporcionar un esquema de codificación sistematizado para ser aplicado en los sistemas de información de la salud.

De esta forma se han analizado diferentes modelos desarrollados bajo diferentes enfoques en los últimos años, teniendo en cuenta las capacidades modeladas y los grupos de referencia de este documento. La Figura 2 muestra la categorización de primer nivel del documento CIDDM.

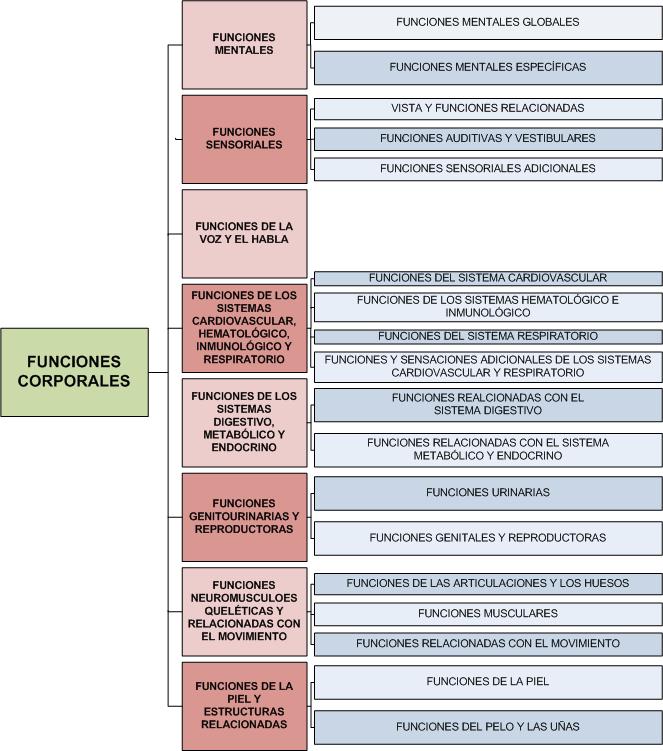


Figura 5. Clasificación de las características corporales.

Una vez estudiadas las capacidades y las limitaciones de usuarios establecidas de manera oficial por la OMS se ha realizado un repaso por aquellas capacidades que se han tenido en cuenta en sistemas de interacción. Muchas de las anteriores categorías y de las características citadas aparecen en los modelos de usuario realizados durante los últimos años.

|  |  |
| --- | --- |
| **Categorías de capacidades** | **Características de usuario modeladas** |
| Sensoriales | * Visuales   + Campo de visión útil (visión de túnel, pérdida de visión central, y combinación de ambas)   + “Stereopsis” * Auditivas   + Pérdida de audición conductiva   + Pérdida de audición sensoneuronal   + Combinación de ambas |
| Factores ambientales | Características contextuales que afectan a las capacidades del usuario, como luz ambiental, ruido., etc. |
| Cognitivas | * Memoria de trabajo * Lenguaje y comunicación |
| Motoras | * Capacidades de las extremidades superiores * Capacidades de movimiento del grueso del cuerpo |
| Físicas | Sin especificar. |
| Cognitivas |
| Discapacidades |
| Necesidades |
| Entorno |
| Experiencia |
| Usuarios ficticios | Nombre, edad, profesión, educación, situación familiar, impedimentos, experiencia tecnológica. |
| Perfil de usuario | Experiencia del usuario  Interfaz  Audio  Pantalla |
| Estado emocional | * Emociones básicas * Felicidad, ansiedad, miedo, amor, odio, orgullo, vergüenza, ira, disgusto, tristeza, satisfacción, confusión, preocupación, aburrimiento, esperanza… |
| Características | Hablador, asertivo, dominante, callado, reservado, tímido, amable, organizado, tenso, ansioso… |
| Personalidad | Extrovertido, introvertido, sensible, optimista… |
| Estado fisiológico | Ritmo cardíaco, presión sanguínea, dilatación de las pupilas, respiración, temperatura, fatiga… |
| Estrés | Ritmo cardíaco, aperturas de la boca, movimientos de la cabeza, presión del ratón… |

Tabla 3. Resumen de características de usuario modeladas.

A continuación se realiza un análisis de las soluciones de la Tabla 1 agrupando lo que se entiende por contexto en DYNUI y tratando de englobar todas las soluciones anteriores en un conjunto de categorías similares a las ya vistas.

|  |  |
| --- | --- |
| **Categoría** | **Características a modelar** |
| Sensoriales | * Visuales   + Campo de visión útil (visión de túnel, pérdida de visión central, y combinación de ambas)   + “Stereopsis” * Auditivas   + Pérdida de audición conductiva   + Pérdida de audición sensoneuronal   + Combinación de ambas |
| Físicas | * Capacidades de las extremidades superiores * Capacidades de movimiento del grueso del cuerpo * Ritmo cardíaco, presión sanguínea, dilatación de las pupilas, respiración, temperatura, fatiga… |
| Cognitivas | * Memoria de trabajo * Lenguaje y comunicación |
| Experiencia | * Experiencia tecnológica * Interfaz * Audio * Pantalla |
| Estrés | * Ritmo cardíaco, aperturas de la boca, movimientos de la cabeza, presión del ratón… |
| Personalidad | * Extrovertido, introvertido, sensible, optimista… * Hablador, asertivo, dominante, callado, reservado, tímido, amable, organizado, tenso, ansioso… * Emociones   + Básicas   + Felicidad, ansiedad, miedo, amor, odio, orgullo, vergüenza, ira, disgusto, tristeza, satisfacción, confusión, preocupación, aburrimiento, esperanza… |
| Factores ambientales | * Características contextuales que afectan a las capacidades del usuario, como luz ambiental, ruido., etc. |
| Otras | * Nombre, edad, profesión, educación, situación familiar |

Tabla 4. Características de usuario unificadas.

Este análisis ha dado como resultado el siguiente modelo de caracterización, personalización y adaptación, el cual está dividido en tres entidades principales:

## Usuario

Modelar las discapacidades físicas de un usuario ya sean naturales (de nacimiento o adquiridas) o generadas por el contexto supone un reto que únicamente personal especializado en materias relacionadas con la medicina pueden abordar con seguridad. En el caso de DYNUI se ha planteado una alternativa: modelar las capacidades. Esto supone añadir un nivel de abstracción al modelado de capacidades de usuario sin dejar de lado las discapacidades que éstos pueden tener, pero resultando más cómodo y directo para generar un proceso de adaptación adecuado. Además una gran ventaja es la de ofrecer la posibilidad al propio usuario de configurar su modelo.

La Figura 3 muestra el modelo de usuario generado en esta fase, el cual se explica a continuación:

* *Interface*: En esta categoría el usuario es capaz de decidir qué tipo de interacción se ajusta más a sus capacidades. De este modo, un usuario con problemas de visión no precisará de indicar (por sí mismo o con ayuda de un especialista) un grado concreto de visión (por ejemplo, un porcentaje de una dolencia concreta) ni siquiera la dolencia específica que sufre. En lugar de esto se le permite elegir el tipo de interacción que desea realizar. El abanico de posibilidades abarca desde una interacción normal con el Smartphone en cuestión (uso de pantalla táctil) hasta el control por voz o uso de gestos. Evidentemente esto se extrapola tanto a la interacción de input o entrada, mediante la cual el usuario envía de alguna forma órdenes al dispositivo, como a la interacción output o de salida, la cual la forman los resultados generados por el dispositivo debido al proceso de interacción.
* *Display*: Con esta categoría se modela la interacción concreta que el usuario puede realizar con la pantalla del dispositivo. Comenzando con una simple característica (“applicable”) se puede determinar si el usuario tiene algún tipo de discapacidad visual (si se modela como “no” se establece una discapacidad visual, con lo que el resto de configuración del modelo resulta más sencillo y automático). Parámetros como la orientación de la pantalla, brillo, contraste y nivel de aumento de la misma son considerados como preferentes en este grupo.
* *Audio*: Del mismo modo que la interacción por pantalla resulta importante, en ciertas circunstancias o según las capacidades del usuario es necesario considerar parámetros asociados al audio. La idea es la misma que la comentada anteriormente en “Display”.
* *View*: Un aspecto importante se modela en esta categoría. Gracias a las facilidades que ofrece la plataforma Android resulta posible establecer la configuración visual de cada uno de los elementos que se van a enseñar por pantalla. Estos elementos son conocidos como Views o Widgets. Mediante este proceso se posibilita la configuración de una interfaz mínima por debajo de la cual el sistema nunca deberá adaptarse. Por ejemplo, si un usuario con cierto problema de visión establece que un botón deberá tener texto blanco sobre negro y un tamaño específico “x”, el proceso de adaptación no deberá disminuir ese tamaño o cambiar esos colores.
* *Other*: En esta categoría se destaca la experiencia del usuario con tecnología, la cual determinará las instrucciones o cantidad de notificaciones que el usuario puede recibir durante el proceso.



Figura 6. Modelo de usuario.

Con el objetivo de completar este modelo de usuario para cada individuo se ha diseñado una herramienta mediante la cual el usuario es conducido a través de un tutorial que es capaz de capturar las capacidades del propio usuario. Presentando diferentes opciones y utilizando diferentes canales de interacción se consigue realizar una captura de sus capacidades, consiguiendo una configuración de interfaz mínima (ver Figura 4). El objetivo de la tarea presentada en esta herramienta es el de enviar un email. De esta forma, la interfaz final estará adaptada a las capacidades del usuario, pudiendo éste enviar dicho email de forma más sencilla.

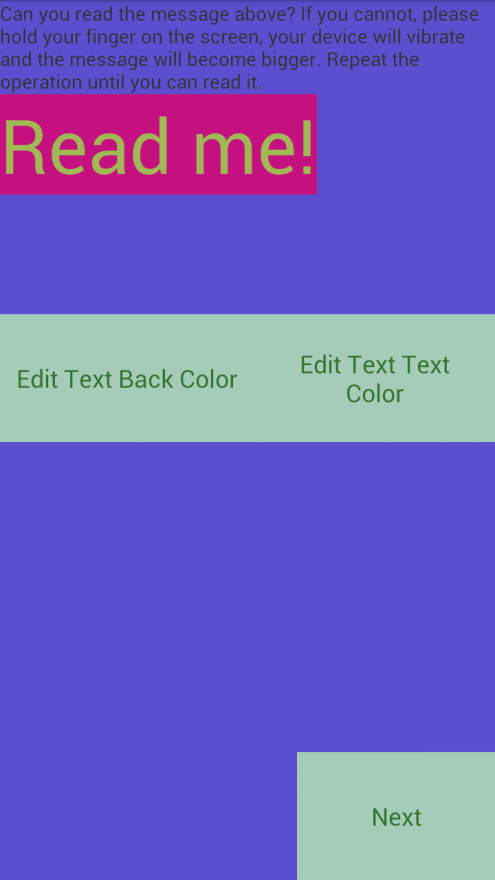


Figura 7. Configurador de IU mínima o modelo de usuario. Tanto el fondo como los tamaños de letra, de botones y textos han sido configurados por el usuario.

## Contexto

El contexto está definido como un conjunto de datos que determinan de manera concreta un entorno. La combinación de información como posición, identidad, actividad y tiempo es suficiente para solicitar otros parámetros que se podrían considerar “derivados” (por ejemplo, datos de contacto a partir de la identidad, o información meteorológica a partir de la posición y del tiempo). La gestión del contexto permite identificar las condiciones de un entorno, de forma que en función de estas condiciones se puedan adaptar los servicios o aplicaciones al usuario. Para ello, es necesario primero la adquisición de la información del contexto, después el procesamiento de esa información recogida del entorno y por último la utilización de la información procesada del contexto para personalizar y contextualizar.

Las variables contextuales son el conjunto de fuentes de contexto cuantificables mediante uso de los distintos dispositivos de sensorización. Sin embargo, el planteamiento de modelado de las variables contextuales difiere con el punto anterior en que no es un acercamiento hardware sino un acercamiento biológico. Es decir, no se plantea realizar un modelado basado en cómo realizar las mediciones de los elementos contextuales sino en qué elementos contextuales se está dispuesto a medir y qué ofrecen esos elementos a la plataforma DYNUI en el objetivo de adaptabilidad perseguido. Por otro lado, además de tratarse de un acercamiento biológico a la situación del usuario también se plantean situaciones que pueden ser deducidas por el sistema o citadas por el usuario y que mejoren el modo en el que se plantea la ubicuidad de la plataforma en el contexto del usuario. En este sentido se contemplan situaciones como la actividad en desarrollo por parte del usuario o el evento bajo el que esté realizando sus acciones de interacción con el dispositivo. Del mismo modo, también se consideran variables contextuales los objetos con los que interactúe el usuario y que puedan proporcionar información sobre el contexto de ejecución de la aplicación.

El contexto se captura mediante sensores. Algunos de los más populares son los siguientes:

* **Ópticos:** Sensores infrarrojos, ultravioleta, sensores de color, foto-diodos… proporcionan información acerca de la intensidad y densidad de la luz, el reflejo, el tipo de luz (artificial o natural)…
* **Audio:** Volumen, tipo de sonido background y frecuencia de base son algunos de los aspectos que pueden ser analizados con sensores de audio y ultrasónicos (especializados en reducir las discapacidades auditivas de los individuos).
* **Movimiento:** Los dispositivos móviles están diseñados precisamente para ser utilizados en movimiento, por lo que capturar la inclinación, aceleración y movimiento en sí del usuario resulta de obvio interés. Los sensores más típicos en este ámbito son los sensores de movimiento angular y acelerómetros.
* **Localización:** Posición, localización, co-localización de otros usuarios y proximidad de usuarios y dispositivos son atributos de contexto importantes en cualquier entorno. Fuera de edificios se utiliza GPS y GSM como herramientas de localización.
* **Biométricos:** Pulso, presión sanguínea… son algunos de los parámetros capturados por este tipo de sensores en dominios de aplicación de medicina. Con procesamiento extra podría extraerse información emocional.
* **Especializados:** Temperatura, presión del aire y sensores similares pueden combinarse para dominios de aplicación especializados.

Algunos de estos sensores han pasado directamente a formar parte del hardware de serie de muchos dispositivos móviles. Este hecho hace posible que la captura de contexto deje de ser exclusiva de entornos inteligentes al uso, siendo posible realizar capturas y ciertas tareas relacionadas en el propio dispositivo. Por ejemplo:

* **Luminosidad:** Muy dependiente del dispositivo a utilizar, el nivel de luz ambiental es una característica fácilmente capturable y muy extendida en el campo de la sensorización móvil. Las pantallas de dispositivos actuales, como smartphones, tienen pantallas conocidas como “glossy” o brillantes. Este tipo de pantallas tienen la capacidad de mostrar colores más vivos haciendo uso de niveles de contraste superiores a las pantallas tradicionales. Además, tienen la ventaja de que aumentan notablemente el ángulo de visión de la pantalla para el usuario. Sin embargo, con ciertos niveles de luminosidad sufre gravemente de reflejos y sombras que dificulta en gran cantidad la interacción. Actualmente la tecnología que se está utilizando en teléfonos de alta gama es la HD Super AMOLED, estando las compañías trabajando en una nueva mejora: Full HD Super AMOLED Plus. Los dispositivos con este tipo de pantallas aportan como solución a este problema una solución software basada en sensores de luz ambiental (ALS). La mayoría de los sensores de luz del mercado actualmente utilizan dos o más tipos de fotodiodos diferentes para medir el nivel de luz, siendo cada cual sensible a diferentes porciones del espectro de luz. Combinando estas salidas de los fotodiodos de forma matemática, cada uno con una ganancia de idoneidad ajustada, el sensor puede generar una medición de brillo ambiental aproximado (http://mobiledeviceinsight.com/2011/12/sensors-in-smartphones/). Utilizando estas capturas como entrada el sistema operativo correspondiente actúa en consecuencia regulando los niveles de brillo de la pantalla del dispositivo en cuestión. Por tanto, vemos que en la actualidad los niveles de luminosidad ambientales ya se están teniendo en cuenta en dispositivos actuales. Las regulaciones de brillo de los smartphones actuales no son más que adaptaciones al contexto actual.
* **Ruido:** Al igual que ocurre con la luminosidad, el ruido (medido en decibelios) es otra de las entradas clave del contexto a tener en cuenta. Un ejemplo muy claro es si pensamos en un lector de libros electrónico. Muchos de ellos poseen la capacidad de leer por nosotros el contenido reproducido haciendo uso de diferentes tecnologías de las llamadas “text-to-speech”. En nuestra casa, con un nivel de decibelios normal, nos servirá una configuración por defecto. Sin embargo, si queremos utilizar el dispositivo en la calle, en el metro, o incluso con un televisor encendido, el contexto cambia claramente y limita las posibilidades de ese dispositivo, así como las capacidades del usuario. Además, hay que tener en cuenta que son cada vez más los dispositivos que cuentan con canceladores de ruido haciendo uso de dobles micrófonos (<http://mobiledeviceinsight.com/2012/01/smartphone-audio-design-dual-noise-cancellation/>).
* **Localización:** La localización establece el punto físico en el que nos encontramos calculado mediante triangulaciones GPS o mediante la conexión a Internet de nuestros dispositivos (resulta menos precisa que la GPS). Esta característica del contexto está muy extendida en ámbitos como el guiado y la recomendación de servicios, ya que los datos extra asociados a las localizaciones son cada vez más ricos.
* **Tiempo y temperatura:** Otra característica muy extendida en el modelado de contexto es el tiempo y la temperatura ambiental actuales. Con ellos numerosas plataformas recomiendan actividades en consonancia con estos parámetros. El principal problema es que se depende de servicios externos para obtenerlos, como Yahoo Weather o Accuweather. De nuevo, y a al igual que la localización, parece claro que la información útil extraíble de esta entrada sería escasa para las tareas de adaptación.

En cuanto a modelos de contexto, se ha revisado la literatura existente comparando las diferentes soluciones. La siguiente tabla muestra una comparación de las técnicas más populares, destacando los enfoques seguidos y las características modeladas en cada caso, así como las categorías en las que se agrupan. La Tabla 4, por su parte, muestra una categorización de características contextuales a partir de la literatura analizada.

|  |  |
| --- | --- |
| **Categoría** | **Características a modelar** |
| Metadatos del entorno | * Localización * Hora |
| Agentes | * Usuario (perfil, localización, preferencias de interacción, entorno y relaciones sociales, etc.) * Dispositivo (capacidad CPU y memoria, interfaces de interacción disponibles, configuraciones posibles de textos, resolución y tamaño de pantalla, conectividad, contenido multimedia soportado, colores, capacidad estéreo, etc.) * Contexto físico (luz, ruido, temperatura, presión, aceleración e infraestructura\*) |
| Entorno virtual | * Actividad del agente * Calendario * Relaciones sociales/laborales |

Tabla 5. Categorías de contexto de DYNUI. La infraestructura contiene información sobre la red, recursos y servicios disponibles, cobertura de red, etc.

|  |  |  |
| --- | --- | --- |
| **Categorías de contexto** | **Características de contexto modeladas** | **Enfoque** |
| Computacional | Conectividad  Costes de comunicación  Ancho de banda disponible  Recursos cercanos disponibles (impresoras, pantallas, estaciones de trabajo…) | Clave-valor |
| De usuario | Perfil de usuario  Localización del usuario  Individuos que rodean al usuario  Localización social del usuario |
| Contexto físico | Luz  Ruido  Temperatura |
| Entidad | Persona  Localización  Objeto genérico |
| Capacidades del dispositivo | Características hardware básicas (potencia CPU, memoria, e interfaz de usuario-teclado, reconocimiento de escritura, voz, o puntero- y salida-pantalla o audio, así como tamaño y resolución de la pantalla, colores, capacidad estéreo, etc.  Características software (contenido multimedia soportado). | Esquema de marcado |
| Características de la red | Ancho de banda, retraso y bit error rate (BER) |
| Información específica del usuario | Preferencias que restringen el acceso al servicio (coste de acceso) y su apariencia (tamaño de fuente de texto, omisión de contenidos, etc.) |
|  | Actividad actual  Actividad planeada  Dispositivos del usuario  Dispositivos o recursos disponibles  Relaciones entre las personas del entorno  Dispositivos necesarios para cada canal de comunicación | Orientación a objetos |
| Conceptos que definen lugares físicos y sus relaciones | Lugar (atómico, compuesto, campus, edificio, habitación, vestíbulo, escaleras, servicios, aparcamiento) |  |
| Conceptos que definen agentes | Agente (persona, agente software)  Rol (orador, audiencia)  Acción intencionada  Acción en presentación |
| Conceptos que definen la localización de un agente en un campus universitario | Entidad en edificio (persona, agente software)  Entidad fuera de edificio (persona, agente software)  Entidad en habitación (persona, agente software) |
| Conceptos que describen la actividad de un agente | Calendario de presentaciones  Evento sucediendo ahora (presentación, habitación ocupada por presentación, participante en presentación-orador o audiencia)  Rol (orador, audiencia) |
| Ontología superior | Localización (interior, exterior)  Actividad (deducida, programada)  Usuario  Entidad computacional (agente, red, servicio, aplicación, dispositivo) |
| Ontología específica |  |

Tabla 6. Tipos de modelos y características de contexto

Como resultado de este análisis se presenta el siguiente modelo:

* *Primary*: Dividida a su vez en físico y de alto nivel, esta categoría se caracteriza por considerar tanto datos atómicos provenientes de sensores como aquellos aspectos de mayor nivel inferidos a raíz de esta información.
* *Secondary*: En este caso, destacan los siguientes parámetros:
  + *Metadatos del entorno*: La información de contexto es normalmente capturada por sensores. Sin embargo, un dato puntual y discreto por sí mismo carece de significado. Rodeando de alguna forma un dato puntual como la hora con otro como el contexto la información se enriquece y cobra mayor sentido. Es decir, un dato discreto como 20ºC por sí mismo no aporta gran conocimiento. Sin embargo, si el dato viene acompañado por un “20ºC en Bilbao” la cosa cambia, ya que no será lo mismo esos 20ºC en Bilbao que en otra ciudad.
  + *Entorno virtual*: Combinando información proveniente de diferentes sensores podemos inferir información de diferentes niveles. Por ejemplo, si la luz de la oficina de Antonio está encendida a las 08:00 de la mañana podemos inferir con cierto grado de probabilidad que Antonio se encuentra trabajando y que ha madrugado. No necesitamos un sensor de presencia o conocer la hora a la que se ha despertado el usuario.
  + *Condiciones de estrés*: Históricamente se han relacionado las actividades del usuario con el modelo del usuario. Sin embargo en este caso para DYNUI entendemos que las actividades vienen determinadas por el contexto actual, el cual puede limitar ciertas capacidades del usuario según la situación.
  + 

Figura 8. Modelo de contexto.

## Dispositivo

En cuanto al modelo de dispositivo cabe sobre todo destacar la consideración que se hace de las características dinámicas del mismo. Es decir, parámetros como nivel disponible de batería, memoria, volumen, etc. resultan importantes cuando se quiere realizar cualquier tipo de procesamiento en el mismo. La Figura 6 muestra la categorización de dicho modelo.



Figura 9. Modelo de dispositivo.

## Diseño del módulo de aprendizaje automático

El aprendizaje de patrones de comportamiento a partir de la información recolectada en la configuración de la aplicación, así como en su utilización, por la complejidad que tiene, demanda una arquitectura dividida en varias capas que al mismo tiempo tendrán varios módulos.

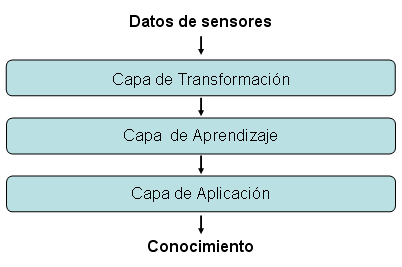


Figura 10 Arquitectura global del aprendizaje

## Capa de transformación

Una vez que los datos hayan sido recogidos, la primera tarea a realizar es inferir acciones relevantes a partir de dichos datos. A veces, la información proveniente será relevante en sí, por ejemplo:

*desde*

*2013-10-20T08:15:57,ButtonBackgroundColor,18*

*inferimos que:*

*2013-10-20T08:15:57, ButtonBackgroundColor,18*

En este caso, la acción en sí es relevante debido a que podemos inferir la acción que el usuario ha llevado a cabo. Hay otras acciones que necesitan ser inferidos a partir un conjunto de acciones recogidos por los sensores. Por ejemplo la acción de “Aumentar el tamaño del botón” no puede ser inferida a partir de una sola información. Para ello, utilizamos un conjunto de acciones simples. Así,

*desde*

*2013-10-20T08:15:54,ButtonWidth,98*

*2013-10-20T08:15:55,ButtonHeight,48*

*inferimos que:*

*2013-10-20T08:15:55, ButtonIncrease,on*

La manera más básica de inferencia de dichas acciones es mediante plantillas que definen qué acciones tienen que combinarse para inferir acciones relevantes. Además de la acciones, se especifican las constraints a cumplir. Estos constraints son relativos a la orden de las acciones como a la duración de las mismas. Por ejemplo, la plantilla para la acción de “ButtonIncrease” es:

*Acciones:*

*ButtonWidth*

*ButtonHeight*

*Constraints:*

*Orden,*

*Tiempo,*

*ButtonWidth (t+1) > ButtonWidth(t)*

*ButtonHeigth (t+1) > ButtonHeigth (t)*

Está claro que la definición de las plantillas está determinada por el conjunto de acciones que queramos identificar así como por la información que obtenemos de cada aplicación y entorno.

## Capa de aprendizaje

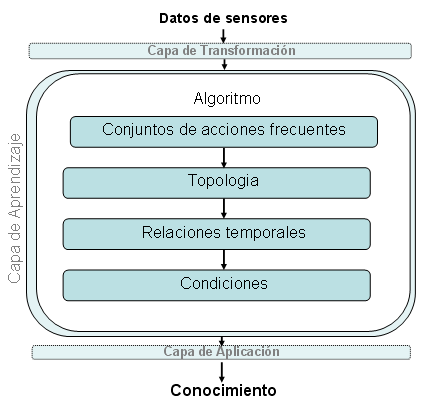
El objetivo de esta capa es la de extraer los patrones de comportamiento más frecuentes a partir de los datos provenientes de la capa de transformación.

Para ello, es necesario definir qué tipo de patrones se quieren descubrir. En este sentido, el objetivo propuesto en este desarrollo ha sido el de:

* Descubrir la configuración preferida por cada usuario en base a los patrones más frecuentes.
* Adaptar los patrones en base a los cambios que realiza el usuario a partir de la configuración inicial.

La idea inicial es el de descubrir aquellos patrones que mejor (o más frecuentemente) definan las preferencias de los usuarios. Además de decidir qué tipo de patrones se quieren descubrir se ha definido cómo se van a representar, ya que la forma en que se quieran representar los patrones influye en el proceso de aprendizaje. Así, se ha decidido representar los patrones mediante secuencias de acciones.

El núcleo de la capa de aprendizaje es el algoritmo que se ha desarrollado. Dicho algoritmo tiene varios módulos que secuencialmente ejecutados descubren los patrones frecuentes. La figura que representa dichos módulos:



### Descubrimiento de los conjuntos de acciones frecuentes

Una vez que los datos han sido recolectados y las acciones de los usuarios hayan sido definidas, el primer paso es identificar qué conjuntos de parámetros son frecuentes en el comportamiento del usuario. Es decir, el objetivo de este primer módulo es la de descubrir qué configuración, definido por los diferentes parámetros, es la preferida por el usuario.

Para ello, se ha utilizado el algoritmo A priori [1], utilizado para el descubrimiento de reglas de asociación. Este algoritmo identifica aquellos conjuntos de parámetros que ocurren más veces que el threshold definido. En este caso, dependiendo de las necesidades de cada entorno, dicho threshold puede variar para encontrar los conjuntos de acciones que se deseen.

### Descubrimiento de la topología

El conjunto de acciones descubierto en el primer módulo define los parámetros preferidos por el usuario, pero no define ningún orden en dichas preferencias. Puede ser que entre dichas preferencias exista alguna relación de orden. Así, el objetivo de esta fase es la de descubrir dicha orden en caso de que existiera.

Para ello, identificamos las secuencias donde se dan dichas acciones y recogemos el orden de las mismas. Basándose en algoritmos de Workflow Mining [2] se plantea la topología base. Aunque los algoritmos de workflow mining son eficientes, no tienen en cuenta ciertos aspectos que son importantes en los entornos inteligentes.

Puede haber un sub-conjunto de parámetros que no tengan un orden definido o que el usuario los haga de forma no ordenada. Para estos casos se ha decidido crear grupos de acciones no-ordenadas. La creación de dichos grupos es definida por varios parámetros que estableces qué relaciones bidireccionales serán considerados como grupos no-ordenados.

### Descubrimiento de las relaciones temporales

En el módulo de descubrimiento de la topología ya se le da una dimensión temporal a la secuencia, explicitando qué acción precede a la siguiente acción. Pero esta relación es sólo cualitativa, es decir, definida solamente mediante relaciones temporales de Allen. En un entorno inteligente, por ejemplo para automatizar la relación entre parámetros, es interesante definir de la forma más precisa posible las relaciones temporales. Así, este módulo trata de descubrir las relaciones temporales más precisas posibles.

Las relaciones temporales se obtienen agrupando las relaciones temporales particulares. Para la agrupación de los valores particulares se utiliza la siguiente fórmula:

**** donde **=** 

Donde: *tolerance* = desviación tolerada de **** (%); ai = la distancia temporal de un elemento; y *n* = número de elementos.

Una vez llevado a cabo el proceso de agrupación, si existe algún grupo que agrupe más instancias que las demandadas, la media de ese grupo será considerada como una relación temporal válida para definir cuantitativamente dicha relación.

### Descubrimiento de las condiciones

La configuración preferida por cada usuario puede cambiar dependiendo de las condiciones. Las condiciones que influyen pueden ser de diferente índole, pero principalmente van a estar relacionados con las características del entorno tales como luminosidad, contraste, acústica etc.

Así, el objetivo es identificar qué configuración prefiere el usuario en cada ocasión. Para ello, la técnica que se utiliza es de clasificación. Por cada configuración se van a guardar en qué condiciones se dio esa configuración, y luego, mediante la utilización de técnicas de clasificación se identifica cuales son las condiciones exactas sobre las cuales se da una configuración u otra.

## Capa de aplicación

Una vez que los patrones ya hayan sido identificados, tales patrones pueden ser utilizados con diferentes fines.

Por un lado, se puede analizar la evolución del usuario en base a los cambios de configuración que realiza. Así, un experto puede analizar dichos cambios e identificar cambios comportamentales que puedan ser significativos.

Por otro lado, y una vez identificado la configuración preferida por cada usuario dependiendo de las condiciones, se puede automatizar la puesta en marcha de las configuraciones. De esta forma, el usuario, cada vez que cambia de condiciones no tiene que adaptar la configuración, sino que se adaptará de forma automática.

## Diseño del módulo de detección de cambios comportamentales

Después de diseñar el módulo de aprendizaje automático, se vio la necesidad de crear un módulo de detección de cambios comportamentales, que comparando el comportamiento actual con los comportamientos detectados por módulo de aprendizaje automático, identifique los cambios comportamentales que se han dado.

Para poder identificar tales cambios se ha desarrollado un algoritmo que teniendo en cuenta el comportamiento actual C = {ci : ci ∈ A}, lo compara con el conjunto de patrones frecuentes fi ∈ F , obtenidos por el módulo de aprendizaje automático.

## Calculando probabilidades

El primer paso para la identificación de cambios es determinar si el comportamiento actual está identificado entre los patrones frecuentes, para ello se verifica que fi ∈ F ∀i. **Si está cubierto por alguna de los patrones, entonces se calcula la probabilidad de tal comportamiento.**

|C0|−1

LL (C|fi)= Pr (ck → ck+1) (1)

k=1

donde |·| denota la cardinalidad del conjunto y Pr (ck → ck+1) es dado por la probabilidad de matriz de transición.

En este caso, si LL(C|fi)=0 ∀ fi ∈ F , indica que el comportamiento actual no está cubierto por un patrón frecuente, de modo que hay un cambio comportamental.

## Calculo de posible modificaciones

Para poder identificar cómo de diferente ha sido el comportamiento, primero hay que identificar todos los posibles caminos que facilita el patrón frecuente, y luego identificar cómo de diferente es el actual comportamiento con cada una de ellas.

La identificación de posibles caminos ha sido desarrollado un algoritmo basado en el algoritmo de busqueda Depth-first search. Una vez que se hayan identificado los posibles caminos, a continuación se compara el comportamiento actual con cada una de ellas, C y ρij,.

Utilizando la distancia de Levenshtein como métrica para valorar los cambios necesarios, el algoritmo identifica qué conjunto de cambios es necesario para que C y ρij concuerden. Los cambios a considerar son:

* H1= {insert(ai): ∀ai ∈ A}: Lo que indica que el usuario olvidó hacer alguna acción.
* H2= {delete(ai): ∀ai ∈ A}: Lo que indica que el usuario ha realizado alguna acción extra.
* H3= {subs(ai,aj): ∀ai,aj ∈ A}: Lo que indica que el usuario ha sustituido la acción ai por la acción aj.
* H4= {swap(ai,aj ): ∀ai,aj ∈ A}: Lo que indica que el usuario realiza las acciones intercambiando el orden de las mismas.

La identificación de las modificaciones que se necesitan se basa en la creación de la matriz de distancia. A continuación se detalla cómo se construye tal matriz.

Algorithm constructDistanceMatrix (C, ρij )

Input: C and ρij

Output: distance matrix (D) y el número de modificaciones number of modiﬁcations

for m =0 to m = |C|for n =0 to n = |ρij|if C(m) == ρij(n) then dm,n = dm−1,n−1

// no necesita modificación

else

dm,n = minimum(

dm−1,n + 1 // insertion

dm,n−1 + 1 // deletion

dm−1,n−1 + 1 // substitution

if((C(m − 1) == ρij(n − 2))&(C(m − 2) == ρij(n − 1)) then

dm−2,n−2 + 1 // swap

)

return D, d|C|,|ρij|

## Diseño del módulo de adaptación del conocimiento

Esta caracterización del contexto viene dada debido a que la propia naturaleza del contexto es tan dinámica que, en ocasiones, es complicado mantener una estabilidad en un proceso de adaptación. Cruzamos calles llenas de gente, entramos en edificios con diferentes grados de luz, usamos el metro, etc. Estas nuevas características se tienen en cuanta para generar normalmente contexto totalmente nuevos. En lugar de esto, lo que pretendemos en DYNUI es mantener el contexto justo anterior modificándolo con las nuevas características encontradas.

La Figura 8 muestra cómo diferentes características del contexto influyen directamente en capacidades del usuario. El ruido, por ejemplo, afecta a nuestra capacidad auditiva del mismo modo que la luz influye en nuestra vista. Pero más allá, el tráfico o la temperatura pueden también perjudicar otras capacidades como pueden ser la movilidad o la precisión de los dedos a la hora de interactuar con una aplicación.

La Figura 9 muestra las relaciones existentes entre las capacidades del usuario y cómo diferentes aspectos del contexto pueden influir en ellas negativamente.

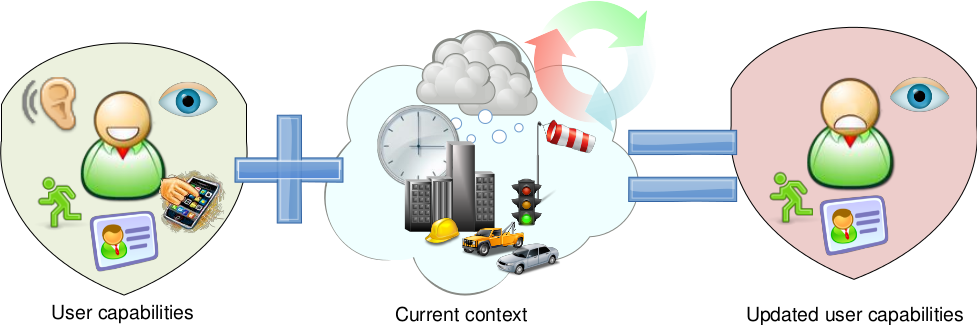


Figura 11. Adaptación de conocimiento del usuario.

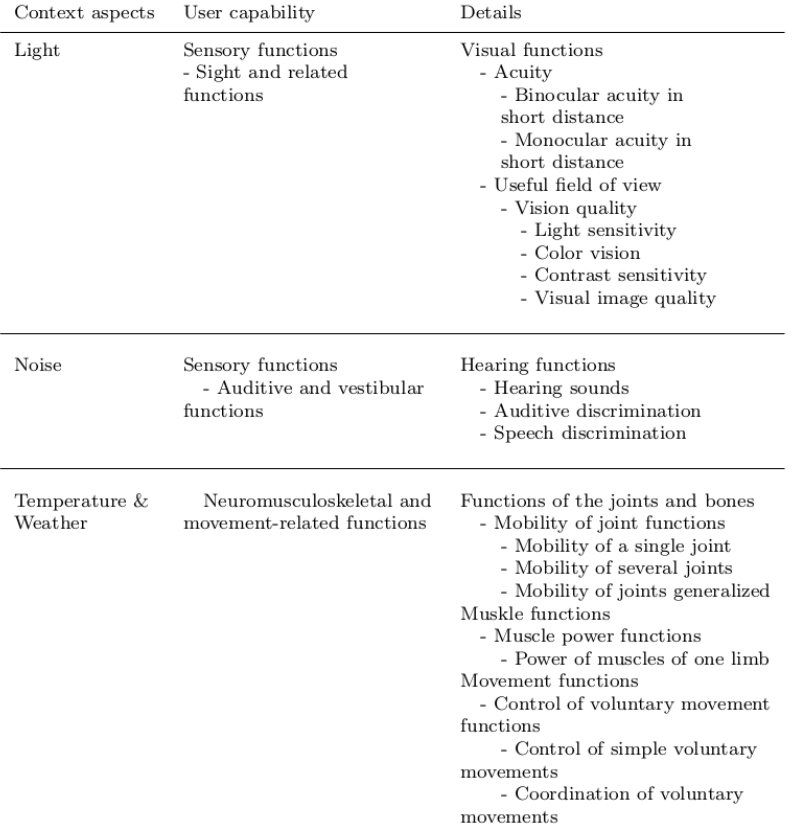


Figura 12. Relaciones existentes entre características del contexto y capacidades del usuario.

## Diseño del módulo para la generación automática de funciones de pertenencia

El objetivo de la generación automática de funciones de pertenencia es la de facilitar la generación de sistemas de lógica fuzzy al desarrollador. En este sentido el objetivo inicial era la de representar algunos de los parámetros, tales como luminosidad, volumen, etc. mediante la representación fuzzy.

Con el diseño del sistema de aprendizaje, si vio que si el sistema era capaz de aprender incluso utilizando datos numéricos en vez de su representación fuzzy, el sistema desarrollado sería más preciso. Así, en un primer momento se ha priorizado la utilización de datos precisos sobre la representación fuzzy.

De todas formas, se ha diseñado el módulo de aprendizaje de tal forma que integre la generación automática de funciones de pertenencia en caso de que fuera necesaria.

## Diseño del módulo de procesado y adaptación

La Figura 10 muestra el proceso de adaptación explicado en el documento de “Uso de tecnologías” y en el primer apartado del presente documento. Los principales módulos que lo componen son:

* *Context Manager*, el cual se encarga de capturar la información de contexto y generar un fichero siguiendo el modelo ya explicado.
* *User Capabilities Reasoner*, módulo que recibe como entrada los modelos de contexto y usuario y genera un nuevo modelo de usuario actualizado según la influencia que el contexto pueda causar en las capacidades del usuario.
* *UI Reasoner*, el cual recibiendo el modelo de capacidades de usuario y dispositivo genera una recomendación para la situación actual.
* *Adaptation Engine*, módulo que comprueba si existe alguna recomendación similar hecha anteriormente y genera la interfaz correspondiente de forma dinámica.

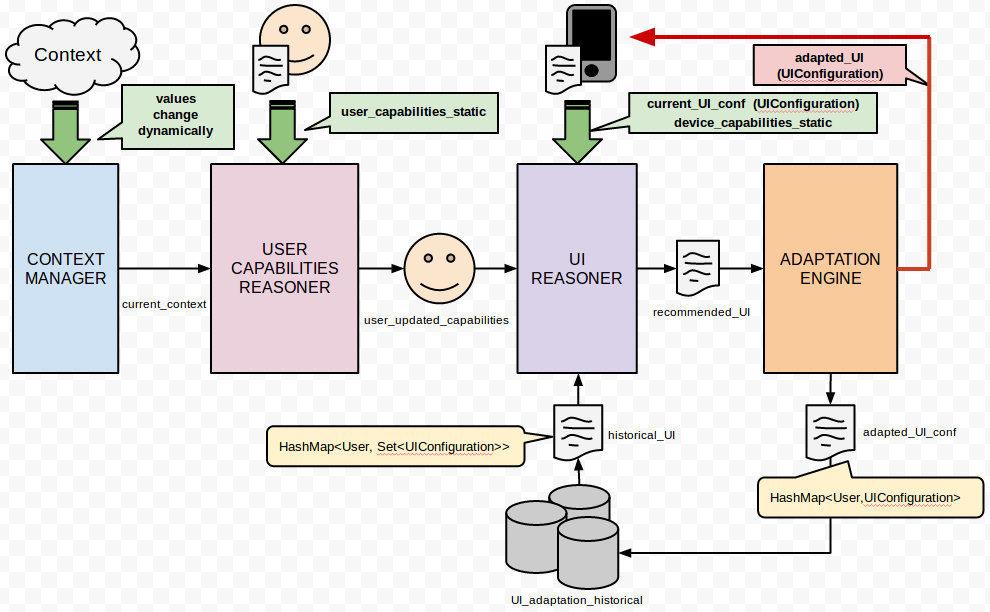


Figura 13.Proceso de adaptación de interfaces de usuario.

A continuación la Figura 11 muestra el modelado inicial de usuarios, dispositivos y contexto con aquellas capacidades que se han tenido en cuenta para la fase inicial de adaptación:

* Usuario: Brillo, volumen, tamaño del “view”, tamaño del texto, salida.
* Contexto: Brillo, ruido.
* Dispositivo: Brillo, volumen, tamaño del “view”, tamaño del texto, orientación.

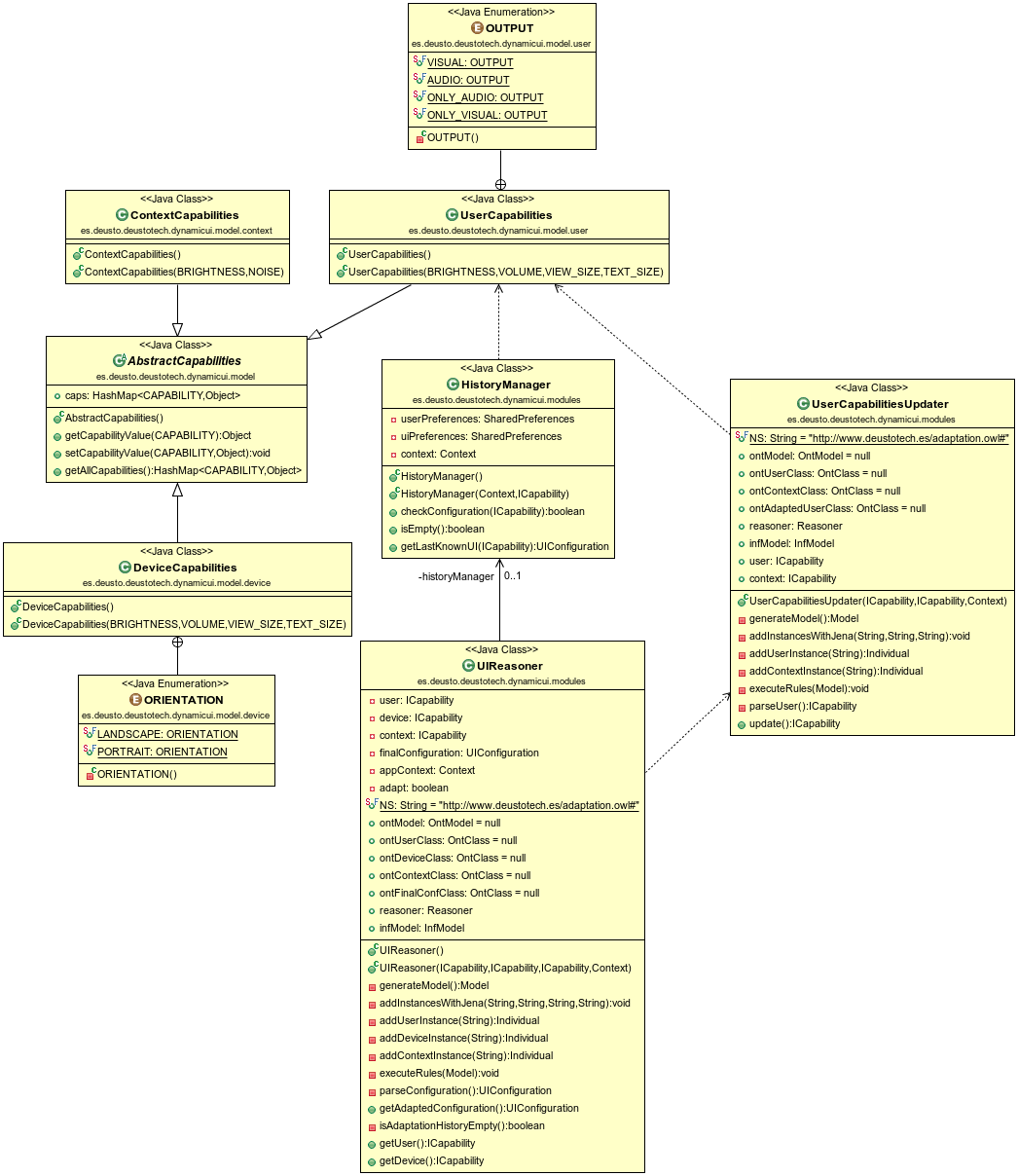


Figura 14. Modelado de usuarios, dispositivos y contexto y relaciones de actualización.

## Diseño del módulo de análisis del comportamiento

Existen situaciones en las que la adaptación no resulta obvia. A estas situaciones las denominamos **incongruencias**. Una incongruencia contextual estará definida por parámetros del entorno que nos inducen a realizar una adaptación. Es decir, nos inducen a razonar de una cierta manera, y a tomar las decisiones correspondientes a las condiciones actuales. Sin embargo, como vamos a estudiar en este apartado, este proceso de adaptación por defecto y lineal no puede por sí mismo darnos respuesta a una situación incongruente. Lo veremos mejor con los siguientes ejemplos:

* Un usuario tiene alguna discapacidad visual que no le permite visionar los contenidos del dispositivo de forma correcta. La lógica nos dice (determinada por el razonador) que la acción correcta a tomar sería la de adaptar la interfaz de usuario a una comandada por comandos de voz, *text-to-speech*, etc. Ahora bien, ¿y si el usuario está en una biblioteca? ¿Y si está en un hospital? ¿O en un examen?
* Un usuario que ve perfectamente, en principio, no debería de sufrir ningún impedimento para interaccionar de una forma normal con su dispositivo. Supongamos ahora que va conduciendo. ¿Dejaríamos que el usuario utilizase su aplicación del mismo modo, de una forma táctil?
* Otro usuario que, estando en su casa, y sin sufrir ningún tipo de discapacidad visual, utiliza su dispositivo de forma normal. Pongámosle en la siguiente situación: son las 13:00 horas y está haciendo la comida. El dispositivo exige su atención. ¿Dejamos que desatienda la cocina?

Si nos fijamos, la solución a estos casos sería una adaptación similar, utilizando las interfaces de audio disponibles. Sin embargo, el problema está en el contexto en sí. Siendo contextos muy diferentes la adaptación sigue siendo la misma, lo cual puede resultar incongruente. ¿Cómo podemos afrontar este problema?

Hasta ahora hemos hablado de agentes. Los agentes han estado formados por el usuario, el dispositivo, y el contexto que les rodea. Llegados a este punto resulta que con estos agentes no es suficiente para caracterizar una situación y tomar las decisiones adecuadas. Las medidas ambientales, por ricas que sean, resultan insuficientes para caracterizar el entorno. Es por ello que debemos introducir el concepto de **actividad**. Las actividades nos ayudarán a entender de mejor forma la situación del usuario y su dispositivo en el entorno. Es decir, enriquecerá el contexto de trabajo. Tener las manos ocupadas, estar en un lugar determinado (como una biblioteca)… son características que deberíamos tener en cuenta a la hora de modelar un contexto. Para ello en DYNUI se propone un modelo de contexto que, además, especifique unos requisitos específicos de la situación actual. En realidad podemos entenderlo como una “humanización del contexto”, ya que lo que vamos a hacer es especificar que en la situación actual la capacidad del usuario va a ser puntualmente diferente.

Estas limitaciones vendrán definidas por cada grupo de actividades. Para ello deberemos identificar un grupo de actividades concreto y **clasificarlas según las restricciones de capacidad del usuario** que generan. Por ejemplo, estar conduciendo o cocinando son actividades que restringen de igual forma al usuario. No nos importan únicamente sus capacidades, sino que además de estas en este caso sabemos que no podrá utilizar la interfaz táctil del dispositivo. Es decir, la **actividad le incapacita puntualmente**. Por tanto, el razonador deberá ser capaz de tomar la decisión de adaptación teniendo en cuenta estos parámetros (no olvidar que, si en estos casos, hubiera una situación de ruido elevado, la adaptación podría no llevarse a cabo). Una primera clasificación podría ser la siguiente:

* Actividades que limitan el uso de las manos.
* Actividades que limitan el uso de la interfaz por voz, ya sea como entrada o como salida.
* Actividades que limitan la vista del usuario.
* Actividades que limitan la atención del usuario.
* Combinaciones de las anteriores.

## Implementación del Módulo de aprendizaje automático

El módulo de aprendizaje se ha desarrollado, para facilitar la integración con los otros módulos, en Java. Para la implementación se ha utilizado el sistema LFPUBS [3] desarrollado para la identificación de patrones frecuentes.

Para la implementación inicial se han tenido en cuenta los parámetros utilizados en la configuración de la aplicación. Así, los parámetros que se han tenido en cuenta son:

* StartTime
* EndTime
* backgroundColor
* buttonWidth
* buttonHeight
* buttonBackgroundColor
* buttonTextColor
* textEditSize
* textEditBackgroundColor
* textEditTextColor
* brightness
* Volume

Así, la implementación del módulo de aprendizaje ha valido para identificar las relaciones frecuentes entre varios de estos parámetros. La transformación para identificar parámetros relevante no ha sido necesaria debido a que los parámetros iniciales eran ya de por sí relevantes.

En la fase de identificación de parámetros frecuentes sí que se han identificado las relaciones frecuentes. Así, diferentes parámetros y sus valores más frecuentes han sido extraídos.

La fase de identificación de topología no ha sido necesaria en este caso debido a que el orden de los parámetros no es crítico en este caso.

Finalmente, sí que ha sido necesario la identificación de condiciones, sobre todo debido a que diferentes condiciones exigen diferentes configuraciones.

## Referencias

[1] R. Agrawal and R. Srikant, "Mining Sequential Patterns," pp. 3-14.

[2] A. W. W. van der Aalst, and L. Maruster, "Workflow mining discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering,* vol. 18, pp. 1128–1142.

[3] A. Aztiria, J.C. Augusto, R. Basagoiti, A. Izaguirre, D.J. Cook, “Learning Frequent Behaviors of the users in Intelligent Environments”, IEEE Transactions on Systems, Man and Cybernetics: Systems, Nov 2013;Vol. 43, Num. 6, pp. 1265-1278