***Ecosistemas de Interfaz Adaptativos***

Antonio Juan Sánchez Martín

Departamento de Informática y Automática, Universidad de Salamanca

[anto@usal.es](mailto:anto@usal.es)

Sara Rodríguez González

Departamento de Informática y Automática, Universidad de Salamanca

[srg@usal.es](mailto:srg@usal.es)

# Resumen

El proceso de diseño de una interfaz es a menudo un proceso largo en que se ven involucrados, al menos, un diseñador de UX y los usuarios. Un Ecosistema de Interfaz Adaptativo(EAI) trata de resolver los conflictos relacionados con el uso de una aplicación de forma particular, es decir, cuando el perfil de los usuarios de una aplicación es tan diverso que la interfaz no puede adaptarse a las necesidades de todos ellos. Generalmente se piensa que una aplicación debe presentar una UX que se adapte a la mayor parte de posible de sus usuarios, y esta interfaz suele variar a través de las versiones de la aplicación modificándose de acuerdo al uso y las necesidades de los usuarios. Un EIA es una interfaz que analiza los patrones de uso del usuario y muta para adaptarse a ellos sin necesidad de ser modificada ni actualizada.

# Motivación

En la actualidad la variedad de interfaces con la que un usuario puede encontrarse ha ido aumentando a medida que se han ido desarrollando y mejorando los sistemas de cognición computarizada.

A día de hoy tenemos ya interfaces visuales, auditivas y táctiles y es fácil que en futuro podamos utilizar todo el rango de sentidos del ser humano para comunicarnos con y a través las computadoras (¿veremos algún día los washapps de sabores?). La idea central de este estudio es la de utilizar los puntos en común que tienen todos los tipos de interfaz y utilizarlos para diseñar un sistema que proporcione la forma de crear interfaces que se adapten, no a un conjunto amplio de usuarios, si no a cada usuario de forma personalizada y automática, basándose en su comportamiento.

# El ecosistema

El hecho de que las interfaces de usuario hayan sido consideradas como entes estáticos a lo largo del tiempo, parte de la propia naturaleza de lo que ha sido considerada una interfaz en el mundo de las comunicaciones durante las últimas décadas, un buen ejemplo de esta consideración es la definición de GUI recogida en ‘The Linux Information Project” [Linux 2008]: *A graphical user interface (GUI) is a human-computer interface (i.e., a way for humans to interact with computers) that uses windows, icons and menus and which can be manipulated by a mouse (and often to a limited extent by a keyboard as well)*. Ha día de hoy la importancia del buen diseño de una GUI viene incluso tipificada en normas ISO (Véanse la normas ISO 9241, donde se estandarizan los principio de la ergonomía). En 1987 la propuesta de un UIDE (User Interface Desing Environment) regularizó el diseño de interfaz de acuerdo los últimos avances tecnológicos de la época, modelando los objetos de datos de las aplicaciones, acciones y condiciones previas y posteriores asociadas con las acciones de los usuario, pero sin tener en cuenta la existencia de tipos de usuarios y sus comportamientos [Foley et *al.* 1990], siempre pesando en la existencia de un “único tipo de usuario”, aplicando concepto como “user-frendly” o “self-explanatory”. En la mayoría de casos a di de hoy estas propuestas siguen siendo válidas puesto que las aplicaciones informáticas tienen usuarios finales muy definidos.

Básicamente una interfaz es un punto de contacto entre dos zonas de mecánicas distintas que requieren comunicarse, son captadores y proveedores de datos, traductores, al fin y al cabo. Pero, por ello mismo, si la mutación y expansión de los lenguajes es algo usual ¿por qué por naturaleza las interfaces son rígidas, y no son ellas mismas las que modifican sus procesos de captación y proyección de datos con el tiempo? En la naturaleza las interfaces se modifican. Un ser humano es capaz de generar lenguaje corporal, los gestos o el tono de su conversación dependiendo de quien tenga delante, ya sea niño o adulto.

Pensemos que una interfaz de usuario es como hábitat natural, donde sobrevive sólo el más fuerte, el más adaptado. Pensemos también, que pese a que el diseñador de cualquier aplicación crea que es lo mejor para el usuario, bien basado en su experiencia o bien basándose en las pruebas empíricas y estadísticas que avalan su teoría, el usuario es un ser autónomo con una forma de proceder, actuar y pensar propia. Añadamos otro pensamiento a nuestra teoría: el usuario es parte de una interfaz de usuario, pero en ella no es parte integrante como un sujeto vivo dentro de un ecosistema, sino que es más un elemento atmosférico, una fuerza natural a la que los elementos del sistema deben adaptarse para fortalecer el sistema mismo y hacerse fuerte con el paso del tiempo. Bien, ya estamos en el punto de partida, bienvenido a los ecosistemas de interfaz adaptativos (EIA).

Si cada elemento de una interfaz es considerado de cómo un individuo que se está adaptando a un sistema, está claro que en su mecanismo debe llevar implementado la funcionalidad necesaria para reconocer los eventos del sistema y, con un objetivo claro, cambiar su forma de proceder para hacer que el sistema en su conjunto sea más capaz de afrontar la próxima eventualidad con mayor diligencia.

La evolución natural es lenta, normalmente hacen falta generaciones de individuos para que una especie altere su estructura genómica y no siempre la mutación desarrollada por los individuos implica que ésta sea un progreso para su especie. A menudo es todo parte de una estrategia de ensayo y error. Si cada especie pudiera acelerar su proceso de adaptación, tanto en tiempo como en acierto de sus mutaciones, obviamente aseguraría más rápidamente su supervivencia y optimizaría el uso de los recursos que tiene a su alcance. Por otro lado, si una especie no está indicada para al entorno (¿Qué va a hacer una foca en el desierto?, prolongar su vida allí sería alargar su sufrimiento) lo mejor es que minimice el uso de recursos o que, si no hubiera más remedio, desaparezca.

Esta analogía con la naturaleza, nos determina ya algunas de las propiedades que deben tener los elementos de un EIA:

**Memoria**. Cada componente debe evolucionar con el tiempo, ello implica que debe tener consciencia de su estado y anotar si las mutaciones producidas tienen consecuencias positivas o negativas.

**Comunicación**. Debe poder comunicarse con el resto de componentes, puesto que las medidas que tome sobre su propia capacidad de respuesta no tienen sentido si puede establecer una escala. Por ejemplo, un botón no podría determinar si una pulsación sobre él, por cada vez que el usuario visita la pantalla que lo contiene, es poco o mucho, sin saber si sobre el resto de componentes se hace alguna acción.

**Límites**. De alguna forma cada componente debe saber en qué rangos debe poder modificarse (no queremos que una caja de texto inunde toda la pantalla o que un enlace sobre política de privacidad desaparezca de una web porque nadie acceda su contenido).

Por otro lado, existe una gran diferencia entre los componentes de una interfaz y los sujetos de un determinado ecosistema: mientras que los segundos tienen como objetivo primario la supervivencia propia y como segundo el de su especie [Freeman, 1977], los primeros deben encargarse de que la interacción que finaliza en la obtención de datos, se produzca de la manera más rápida y cómoda para el usuario.

# Las mutaciones

Una mutación en la naturaleza es “un cambio en la secuencia genética, y es la causa principal de la diversidad entre los organismos” [Loewe, 2008]. ¿Cómo trasladar ese concepto al mundo digital? El concepto de mutación digital no es nuevo, lo encontramos, por ejemplo, en los virus. La diferencia entre las mutaciones de los virus y las de los elementos de una interfaz, es que mientras los virus pueden cambiar la estructura de su código (para que al replicarse sea más difícil de ser localizado por los antivirus) los elementos de una interfaz deberán modificar su apariencia externa y cuando lo requieran también su funcionalidad.

Las mutaciones biológicas de una especie propician efectos colaterales: si una especie se adapta mejor al medio y consigue prosperar, a menudo es usual que esto lo haga a costa de otras especies. En la naturaleza este proceso se autorregula con el tiempo y son predecibles gracias teorías de dinámicas de sistema como las descritas por las ecuaciones de Lotka–Volterra. Por desgracia en un entorno digital las autorregulaciones no existen, y los límites de las mutaciones deberán ser moderados previamente al inicio del sistema (véase la sección de ‘Límites’).

¿Qué parámetros (visuales y funcionales) de un elemento de interfaz son susceptibles a mutaciones? Considerando la categorización más simple posible entre elementos de una interfaz (elementos de entrada, elementos de salida, elementos de entrada-salida y elementos compuestos) podemos reducir los factores de mutación a mutaciones de pregnancia[[1]](#footnote-1) y mutaciones de comportamiento. No ha de confundirse la mutación del comportamiento con la funcionalidad última del sistema asociada la interfaz, sino con la forma misma de actuar de la propia interfaz.

Para explicar un poco mejor lo referente a mutaciones de funcionalidad abordaremos el siguiente ejemplo: Un usuario accede frecuentemente a la funcionalidad asociada a un segundo nivel de menú de una aplicación, de hecho, lo hace más frecuentemente que algunas de las opciones dispuestas en el primer nivel de dicho menú. La mutación normal en este caso sería la inclusión del elemento del segundo nivel de menú dentro del primer nivel, en detrimento de alguno de los elementos de primer nivel de menú menos utilizados. Pueden considerarse también como mutaciones de funcionalidad la preselección de elementos o los cambios en las sugerencias de datos de entrada.

Como se ha comentado previamente, las mutaciones de un elemento de una interfaz difieren en su intención última frente a las mutaciones de la naturaleza. Las mutaciones biológicas tienen como objetivo mejorar la adaptación del individuo al medio, llevándole a una posición más favorable, las mutaciones digitales en este caso no favorecerían al individuo sino al usuario final de la interfaz. Esto nos lleva a considerar que las mutaciones no serán ‘favorecedoras’ en todos los casos, y que podemos encontrar elementos que pierden prestancia en cuanto a pregnancia o comportamiento para favorecer a otros.

Para que un elemento pueda saber exactamente cuándo mejorar sus condiciones o cuándo retraerse frente al resto de elementos, debe tener consciencia del estado de éstos, por ello el mecanismo de comunicación entre los elementos es muy importante (véase la sección de ‘Comunicación’); pero de igual forma deben tener consciencia de las condiciones del entorno, es decir los recursos disponibles.

Puede parecer evidente considerar que si un elemento gana prestancia dentro de la interfaz a través de una mutación, la disminuya en una cantidad en un grado proporcional para compensar el resto de elemento, de tal forma que la suma total de las pregnancias (*P*) de elementos antes y después de la mutación sea igual, para que el sistema continúe estable.

(Ecuación 1)

Sería algo similar a la ley de conservación de la energía, pero no sería cierto en todos los casos. Si bien el crecimiento de la prestancia de los elementos de un sistema no es infinita, pueden existir estados en los que los recursos de un sistema no se hayan explotado por completo y la suma total de las pregnancias de los elementos sea distinta antes y después de la mutación. Un ejemplo sencillo de este tipo de estados puede ser el siguiente: consideremos una aplicación software con menús, en la que un elemento del menú de segundo nivel es nuevamente más utilizado de que uno de primer nivel y por tanto debe ‘promocionarse’ hacia el primer nivel. El comportamiento esperado en este caso es que uno de los componentes del primer nivel pase a ser del segundo para contrarrestar la mutación, pero esto sólo sería necesario si el número total de elementos de primer nivel hubiera llegado a su tope, en caso contrario únicamente se verían incrementado los elementos de menú de primer nivel.

Utilizar los menús como ejemplo nos puede llevar a pensar incluso que la pregnancia total de un sistema (considerada como la suma de la pregnancia de sus elementos) puede llegar a ser infinita: podemos llegar a tener un número infinito de niveles de menú, pero ello claramente no es lo deseable para una interfaz, ni es tampoco es un caso real que una interfaz tenga infinitas funcionalidades. Por ello, antes del diseño de la implementación de un EIA resulta inevitable la definición de límites.

Como consecuencia de la evidencia de recursos no utilizados de un sistema podemos redefinir la ecuación anterior de la siguiente forma (donde *Pu* es la pregnancia asociada a los recursos no utilizados):

(Ecuación 2)

## Ambientes

Parece evidente, pero es importante señalar que por su naturaleza no todos los elementos de una interfaz tienen prestaciones similares y por tanto los cambios en las pregnancias de los elementos deben afectar solamente a sus competidores directos; es decir no tendría sentido que por ejemplo en una aplicación web el intercambio de elementos de menú entre diferentes niveles afecte al tamaño del texto de las cabeceras o al color de los elementos del pie de página. Por ello, debemos considerar que los elementos de una aplicación se asocian en *ambientes*. Podemos definir un *ambiente* como el conjunto de elementos de una EIA que se ven afectados por la mutación de uno de sus elementos, y así volver a redefinir la función de pregnancia:

(Ecuación 3)

Los ambientes pueden ser considerados también como elementos de un EIA de tal forma que pueden mutar y competir entre sí en caso de ser necesario. No tiene sentido que la mutación de un elemento de menú afecte a una caja de entrada de texto de un formulario, pero sí que el tamaño del espacio asignado al menú se vea afectado si el tamaño del menú que se encuentra encima se incrementa.

Como elemento de un sistema los ambientes pueden también agruparse entre sí dando lugar a nuevos ambientes. Por ello la pregnancia de un sistema viene determinada como la suma de las pregnancias de cada uno de los ambientes de primer nivel (aquellos que no están contenidos en ningún otro ambiente) más la suma de la pregnancia de los recursos no utilizados:

(Ecuación 4)

## Recursos

Solamente tiene sentido definir un recurso dentro de un ambiente, pues será aquello afectado a lo que repercutan las mutaciones de los elementos del ambiente. Los recursos pueden ser bien propiedades estéticas (tamaño, brillo, contraste, tono, saturación, posición, etc.) si estamos tratando con una interfaz visual, o bien propiedades sonoras (tono, volumen, etc.) si se trata de una interfaz sonora. Un *recurso* dentro de una EIA es en resumen una propiedad física que proporciona prestancia a un elemento de la interfaz que pueda ser detectada por los sentidos y mensurable (no se deben por tanto considerar propiedades visuales, la intensidad de un *umami* puede ser considerado también un recurso si consideramos elementos de una interfaz gustativa).

## La memoria

Como todo componente consciente dentro de un sistema, un elemento de EIA debe poseer una memoria que le permita conocer su propio estado y del ambiente a lo largo del tiempo para decidir qué acciones tomar cada vez que tome contacto con el usuario de la interfaz. La memoria debe almacenar por tanto los datos de los recursos consumidos por elementos y por el entorno.

No es necesario que cada elemento almacene los datos de cada uno de sus compañeros dentro del ambiente, pero sí para facilitar los cálculos; como se verá más adelante deberá almacenar los recursos utilizados y disponibles de su ambiente.

Dentro de la memoria cada componente deberá almacenar también los límites que afecten también a sus capacidades de mutación.

## La mutación de prestancia

De inicio, el diseñador de la interfaz considerará que existen elementos de menor o mayor prestancia, puesto que es imposible diseñar un sistema donde todos los elementos sean percibidos por el usuario con la misma consideración, ya solamente su posición hace decantar al usuario hacia una determinada categorización.

¿Cuándo ha mutar un elemento? ¿Cuándo ha de crecer o disminuir la prestancia un elemento? La mutación en prestancia es mucho más fácil de calcular que la de funcionalidad al tener a disposición cantidades mensurables. En el mundo biológico la mutación mínima en procesos de reproducción es de un gen, en el caso digital no hace falta que el elemento se reproduzca para mutar, pero sí que es necesario determinar cuál es la cantidad mínima, es este caso de prestancia, que puede ser alterada [Bundesen *et al,* 2015] en una mutación, con el fin de poder operar correctamente cada vez que un elemento se modifique.

### La unidad mínima de prestancia

Al no ser la prestancia una magnitud física concreta, sino que más bien una suma de cualidades de un objeto, la identificación de una unidad mínima no es trivial. Una unidad mínima de prestancia podría ser un pixel si esta determina el tamaño dentro de una pantalla de ordenador o un decibelio si se trata de comparar sonidos; en resumen, el rango de propiedades que pueden afectar a prestancia de un objeto puede ser muy grande.

Un factor clave para determinar esta unidad mínima es que el incremento o decremento de prestancia entre dos elementos solo puede ser determinado dentro de un ámbito donde ambos elementos puedan ser considerados iguales. Ese ámbito dentro de una EIA será el ambiente. Lo cual determina que *todos los elementos dentro de un ambiente utilicen la misma unidad mínima de prestancia*.

La forma o fórmula de determinar la unidad mínima dentro de un ambiente, se realizará de forma manual, es decir, el diseñador/programador de la interfaz identificará las propiedades que modifican la prestancia de los elementos del ambiente.

Pueden llegar a existir casos en la prestancia sea una unidad compuesta, como por ejemplo la dupla {altura en píxeles, intensidad de color rojo (0, 255)} lo cual puede llegar a complicar los cálculos. Aunque en estos casos es posible también establecer una unidad mínima, como por ejemplo {1px, una unidad de color}, no es lo deseable, aunque sí hay estudios que permiten este tipo de cálculos [Bundesen *et al,* 2015]

## La variación

Si un elemento de una interfaz es más utilizado que otro, lo ideal es que este gane prestancia frente al resto [Gould, 1983]. Bajo esta base y para estar de acuerdo con la (Ecuación 2), el incremento de la prestancia de un elemento(Pe) debe ser igual a la suma de los decrementos de las prestancias del resto de los elementos más el decremento de la prestancia no utilizada (asociada a los recursos no consumidos):

(Ecuación 5)

De tal forma que

(Ecuación 6)

### La velocidad de mutación

La mutación no es por necesidad algo que se produzca cada vez que se produzca una interacción con el usuario. En los ciclos biológicos hacen falta generaciones para que una especie mute (aunque esto está condicionado fundamentalmente porque las mutaciones se producen durante el proceso reproductivo) y aunque esto no es necesario en un entorno digital, lo cierto es que un análisis más contrastado de las condiciones de uso de la interfaz llevará posiblemente a una comprensión mucho mayor de los procesos de interacción del usuario. Por ello, generar una mutación instantánea cada vez que se produzca una interacción con la interfaz no es lo más indicado.

Lo ideal es que elementos y ambientes utilicen su memoria para almacenar durante un periodo de tiempo las interacciones del usuario y luego realizar estadísticas de utilización antes de tomar una decisión.

La velocidad de adaptación del sistema será algo que puede variar a lo largo del tiempo. Lo más habitual es que la interfaz varíe muy rápidamente al principio y que luego pase a largos periodos de estabilidad, de igual forma que pasa con los sistemas biológicos. El método de aprendizaje de un perceptrón simple (Rosenblatt, 1962), donde los pesos varían muy rápidamente al principio de su aprendizaje puede ser resultar un modelo muy similar al esperado.

El concepto velocidad de mutación inicial, que denominaremos vm será muy similar al de momentum de una red multicapa y será una constante en la función de mutación.

### La función de mutación (τ)

La decisión a la hora de qué elementos mutar o cuál será su transformación en unidades mínimas de prestancia deberá ser tomada atendiendo a los datos almacenados en la memoria acerca del uso del elemento por parte del usuario. Cada elemento deberá definir qué representa para el mismo su utilización (un click, el completado de un campo, la aceptación de una orden sonora… dependerá de la naturaleza del elemento y del tipo de interfaz).

La variación de la prestancia de un elemento será determinada por la siguiente ecuación:

(Ecuación 7)

Mientras, la disminución de la prestancia del resto de elementos se calculará de la siguiente forma:

(Ecuación 8)

La variación de la prestancia de los recursos no utilizados se calcularía de igual forma

(Ecuación 9)

## La mutación de funcionalidad

Las mutaciones de funcionalidad son fácilmente detectables por humanos, pero es difícil de encontrar una solución que resuelva la globalidad los casos. La mayoría de veces las mutaciones de funcionalidad pueden reenfocarse como mutaciones de prestancia, reevaluando las directrices que determinan las funciones de mutación de cada ambiente.

Un ejemplo claro de mutación de funcionalidad es la que agruparía el resultado de varios procesos de cálculo. Este caso podría solucionarse si a la hora de planificar la funcionalidad asociada a la interfaz se decidiera incluir un elemento que realizara el cálculo completo, y este elemento poco a poco a través de mutaciones de prestancia fuera adquiriendo una posición más relevante con relación a su uso.

# La comunicación y memoria

En una EIA, la comunicación entre los elementos que la forman es providencial a la hora de plantear el momento y la intensidad de las mutaciones. Cada elemento debe tener constancia de forma pormenorizada de su propia interacción con el usuario y forma generalizada de la relación del usuario con el resto de elementos del ambiente.

En cada interacción el elemento debe modificar los datos acerca de las interacciones y el ambiente almacenados por el mismo y comunicar al resto de elementos que se ha producido una interacción. Si el ambiente al que pertenece el elemento es parte a su vez de otro ambiente, debe realizar la misma operación. De esta forma cada interacción del usuario supondrá una sucesión de notificaciones en cascada desde el elemento con el que se interaccione hasta los elementos que componen el ambiente de ‘primer nivel’ del sistema (ver Imagen 1)

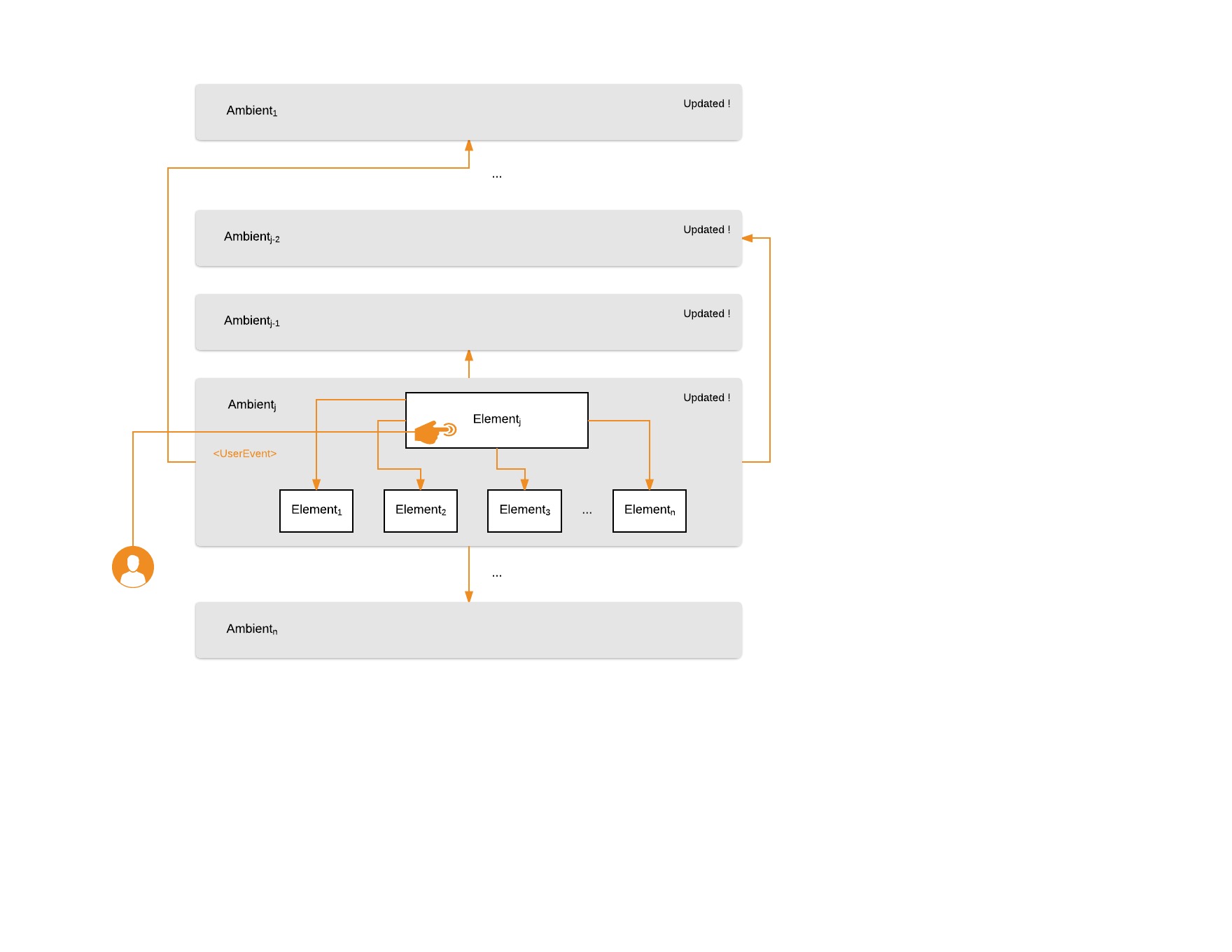


Imagen 1. Notificación de interacciones

Una vez que todos los elementos están actualizados, se procede a evaluar si han de producirse mutaciones. Eso implica que el proceso de actualización debe notificar a quien lo inicia que ha acabado (ver Imagen 2)

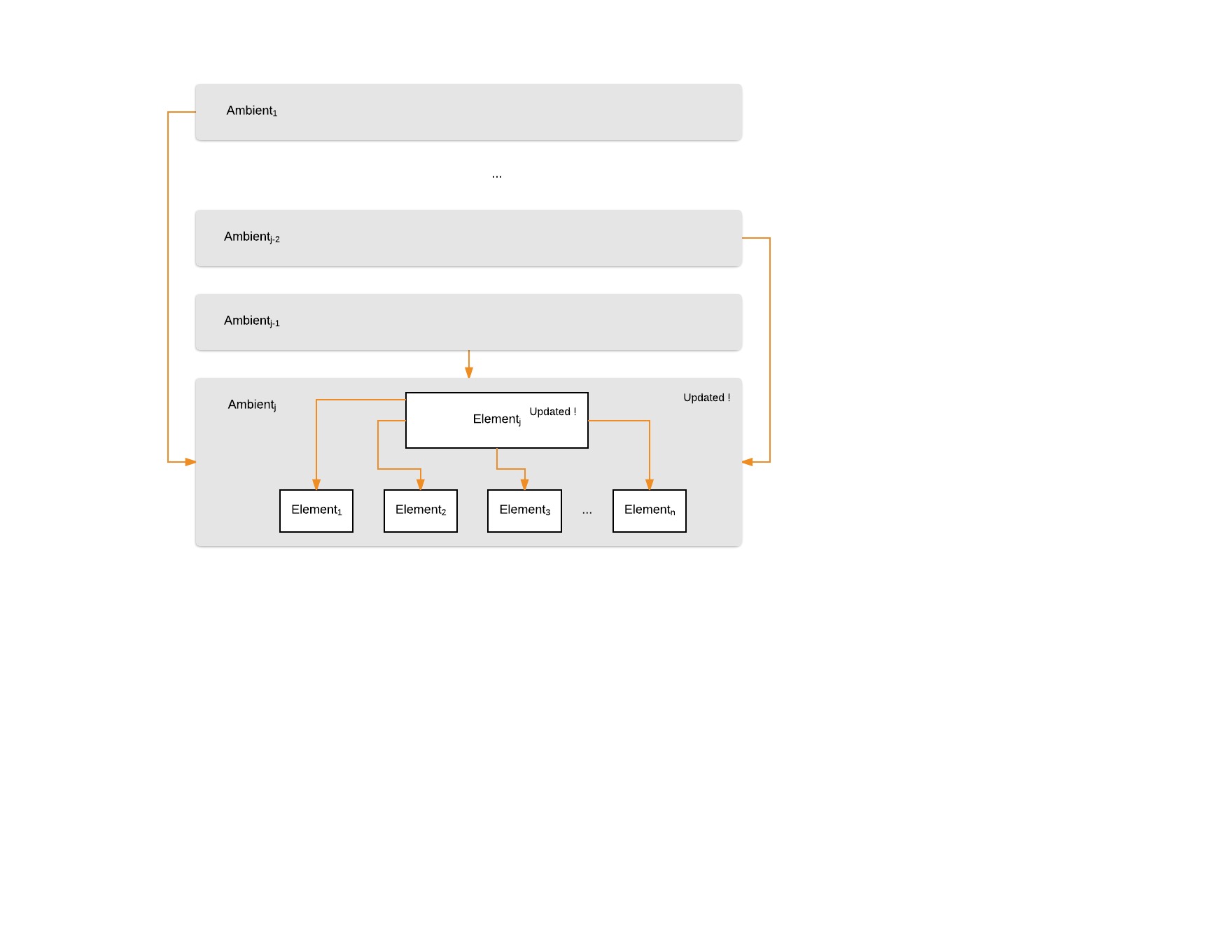


Imagen 2. Notificación de interacciones

Los cambios provocados por una mutación no deben afectar a ningún elemento fuera del ambiente del que se produzcan, tanto si los recursos del sistema están siendo completamente aprovechados como si no.

Para tener control de los recursos no utilizados dentro de cada nivel, deberá existir un componente que los controle. Este componente puede ser o bien el propio elemento ambiente o bien otro componente aislado, dependerá de la implementación.

Como es posible tener interacciones de forma concurrente en el sistema, lo ideal es que exista un mecanismo de comunicación en el que se puedan encolar los mensajes sobre las interacciones y los mensajes sobre los procesos de mutación que se lleven a cabo. Para que unos no interrumpan a otros ni los cálculos se basen en datos desactualizados (ver Imagen 3).

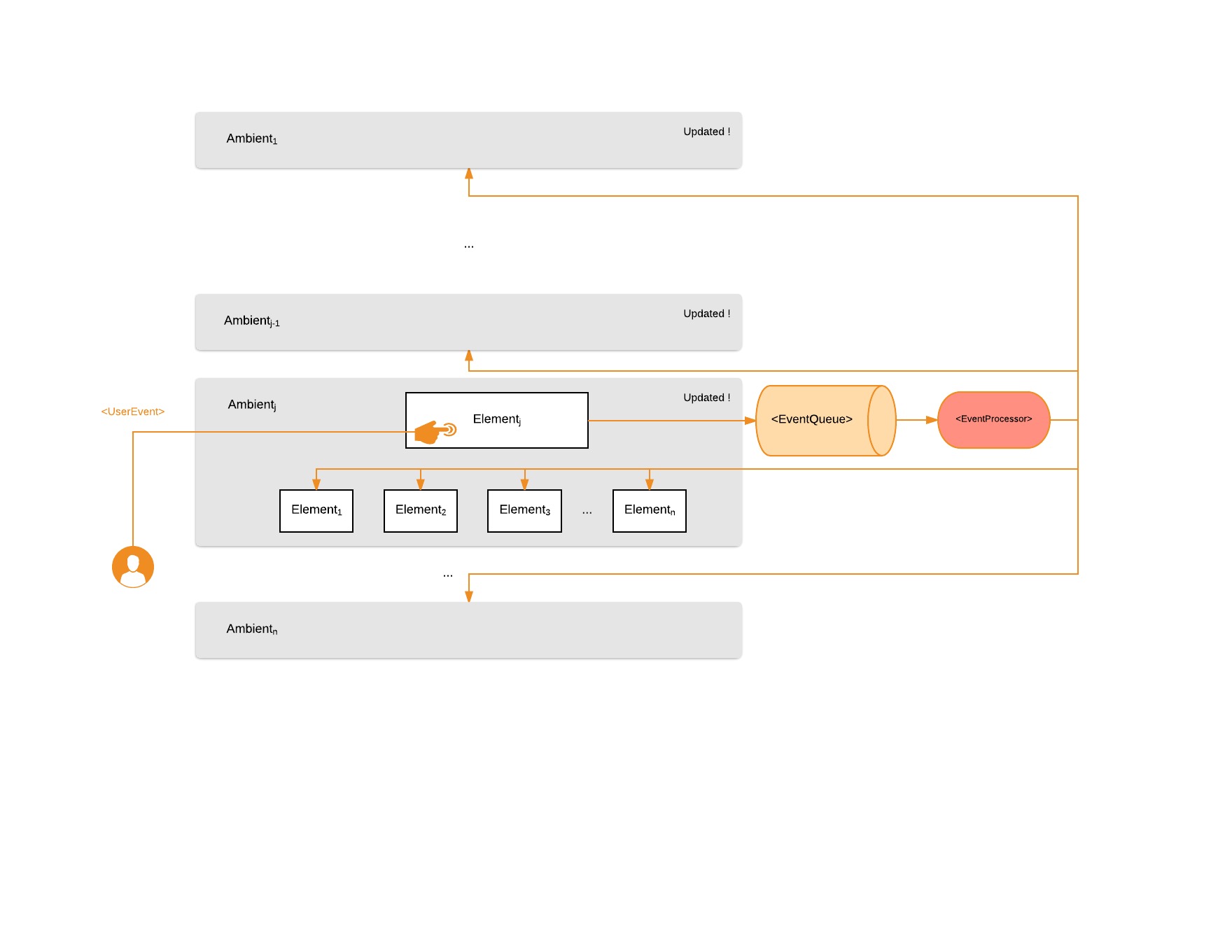


Imagen 3. Encolado de eventos

# Configuración y límites: IEA y diseño de UX

Como se ha mencionado anteriormente la formalización de límites es una tarea prioritaria en el diseño de una EIA. Antes de la entrada en funcionamiento de un EIA se ha de definir la prestancia de cada elemento y de los recursos disponibles, la velocidad de mutación y la función de mutación.

El hecho de que los elementos de la interfaz muten libremente puede entrar en conflicto con las consideraciones estéticas (estéticas en su más amplio rango, no solo visual que es el más comúnmente tratado). Por ello el establecimiento de los recursos disponibles de una EIA no tiene porqué ser el del total del sistema, esto debe determinarlo un diseñador de UX. Los límites de crecimiento y decrecimiento de los elementos del sistema deberán ser establecidos por el diseñador de la interfaz de este, de tal forma que el aspecto estético no se vea afectado ni se invada el espacio de otras funcionalidades del sistema a los que no aplique un IEA. Este tipo de decisiones, que son del tipo: tamaño máximo de las cajas de entradas del sistema, tamaño mínimo de los textos o de las imágenes o volúmenes máximos y mínimos de los sonidos del sistema, son las cotas que debe establecer el diseñador de UX a la hora de implantar un sistema IEA.

Las funciones de mutación se han de definir pensando también en que sean una forma de limitar la aparición de generaciones en lapsos de tiempo tan pequeños que al usuario le cueste asimilar los cambios. Por ello las funciones de mutación no han de ser necesariamente continuas.

# Actualizaciones del sistema

Una duda que surge al ver un EAI es si esta debe reiniciar sus cálculos si un elemento nuevo aparece en el sistema. Por ejemplo, supongamos que en una aplicación de móvil aparece un nuevo elemento asociado a una nueva funcionalidad, ¿que pasaría entonces con ese botón? ¿se incluiría en el sistema con el valor mínimo para su prestancia? ¿si es una funcionalidad nueva podría considerarse que al usuario le sería de mucho interés conocerla y por tanto debe incluirse en el sistema con un valor máximo para su prestancia?

Está claro que la importancia inicial de este nuevo elemento no viene definida ni por el mismo, ni por el sistema, sino que es el diseñador de UX el que debe decidirlo.

Podemos encontrarnos 2 casos diferenciados en lo que refiere a la inclusión de nuevos elementos en el sistema: el primer caso es que elemento sea totalmente nuevo y no tenga relación con el resto de elementos del sistema (por ejemplo un elemento de menú asociado a una nueva funcionalidad que no tiene ninguna relación primaria ni asociada con el reto de funcionalidades del sistema –imaginamos una app para dispositivos móviles de gestión correo electrónico que permitiera como novedad gestionar el envío postal tradicional y que tuviera como consecuencia un elemento nuevo en el menú principal que diera acceso a esa funcionalidad), en este caso el diseñador de la UI debe poder precisar el lugar exacto en el que debe colocar el elemento de menú.

El segundo de los casos es que el nuevo elemento esté directamente relacionado con uno de los elementos ya presentes en el sistema y por ello puede asumirse que la prestancia de ese elemento será la misma que la del elemento ya presente.

En ambos casos el diseñador de UX debe decidir que prestancia inicial signarle al nuevo elemento, si una asociada o una nueva. La asignación directa de una prestancia puede solucionar esta problemática. Pero también puede abordarse desde una solución más técnica y es la de inclusión del tiempo de vida de un elemento en la fórmula del cálculo de la prestancia, viendo esta así definida por 2 parámetros:

(Ecuación 10)

El primero como ya se ha definido anteriormente sería el número de interacciones del elemento frente al del ambiente, el segundo sería el tiempo de vida del elemento frente al tiempo de vida del ambiente (medido en ciclos). Esto nos llevaría a introducir un nuevo concepto el de ciclo de actualización.

**Ciclo de actualización**. Número de veces que un elemento o ambiente del sistema sufre una actualización. La memoria del sistema debe llevar cuanta de estos ciclos. Que se incrementan cada vez que se produce un evento dentro del sistema que conlleve una mutación de la UI.

Definidos los ciclos tenemos que:

(Ecuación 11)

donde ti es el número de ciclos del elemento y tambient el número de ciclos del ambiente que lo contiene.

# La función de maduración. El fin de la mutación.

# Es importante definir cuando el ecosistema se ha adaptado al usuario, no es práctico que el ecosistema esté mutando a lo largo de todo su ciclo de vida, ya que en determinado momento se puede producir patrones de comportamiento que se alejen de lo habitual por parte del usuario y eso lleve al sistema a evolucionar hasta un estado no deseado (sería algo similar al overfitting de las redes de neuronas).

# Con el concepto de ciclos ya introducido, una función de control puede hacerse cargo de limitar el periodo de mutación, así como de acrecentar o disminuir su efecto. A esta función la denominaremos función de maduración (ξ).

Un sistema se mantendrá estable mientras haya finalizado la mutación y no hayan aparecido elementos nuevos que puedan provocar al usuario un comportamiento diverso. Por ello el parámetro que debe recibir esta función no es número de ciclos del ambiente, sino el número de ciclos desde la última ver que se ha introducido un elemento en él (o en su defecto desde el inicio del sistema). A este variable, que se entiende volverá a cero cada vez que sea introducido un nuevo elemento lo denominemos grado de maduración del ambiente(m). Así:

(Ecuación 12)

# Ejemplo: mutaciones de una interfaz visual

En el siguiente ejemplo se puede apreciar cómo en una interfaz web sencilla, como puede ser la pantalla de entrada de un browser, la disposición de los elementos a lo largo del tiempo muta hasta resultar mucho más adecuada a los requisitos del usuario.

AAAA -> BBBB -> CCCC -> DDDD

Para ello se ha elaborado un diseño muy sencillo de la interfaz donde la parametrización de los componentes se ha llevado a cabo de la siguiente forma:

Completar

# Datos compartidos

Como se ha podido apreciar en el ejemplo, el uso de esta aproximación reduce de forma significativa el trabajo de un diseñador de UX y además mejora susceptiblemente el manteniendo de un interfaz, puesto que ella misma es capaz de adaptarse a los usuarios que la utilicen de forma individualizada, porque en ningún momento es necesario que los datos de la evolución dxe una interfaz sean compartidos por todos usuarios.

Resulta interesante saber cómo y en qué grado podrían ayudar los datos obtenidos de un conjunto no homogéneo de usuarios a la hora de hacer evolucionar más rápidamente a un EIA.

# Conclusiones

La EIA son aplicables

# Bibliografía

[Foley et *al.* 1990] Foley, J., A. van Dam, S. Feiner, and J. Hughes, (1990) *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading, MA, 1990. ISBN-13: 978-0-321-39952-6

[Bundesen *et al,* 2015] Bundesen, C., Vangkilde, S., & Petersen, A. (11 de 2015). Recent developments in a computational theory of visual attention (TVA). (J. K. Tsotsos, M. P. Eckstein, & M. S. Landy, Edits.) *Vision Research, 116*, 210–218.

[Gould, 1983] John D. Gould, C. L. (1983). Designing for usability - key principles and what designers think. En A. Janda (Ed.), *Conference on Human Factors in Computing Systems* (págs. 50-53). Boston, Massachusetts: ACM New York.

R.A.E. (2006). *Diccionario esencial de la lengua española* (1 ed.). Madrid: Espasa Calpe.

Rosenblatt, F. (1962). *Principles of neurodynamics; perceptrons and the theory of brain mechanisms.* Washington, EE.UU.: Spartan Books.

[20] Myers, B.A. (1998): A brief history of humancomputer interaction technology. ACM Interactions, 5(2), 44-54.

[7] Grudin, J. (1990): The computer reaches out: the historical continuity of interface design. Proceedings of CHI'90, 261-268.

[8] Grudin, J. (2005): Three faces of Human-Computer Interaction. Annals of the History of Computing, vol. 2 no 4, 2-18. [9] Hiltzik, M.

[13] Jørgensen, A.H. (2007): Mainstream versus Mainstream: Two Approaches to User Interface History. Proc. 7th Danish HCI Symposium, IT University of Copenhagen, Nov. 22, 2007, 33-34

[Linux, 2008] GUI definition. Linux Information Project. October 1, 2004. Retrieved 12 November 2008. http://www.linfo.org/gui.html

[Freeman, 1977] Freeman, R. B. (1977), "On the Origin of Species", The Works of Charles Darwin: An Annotated Bibliographical Handlist (2nd ed.), Cannon House, Folkestone, Kent, England: Wm Dawson & Sons Ltd

**Loewe, L. (2008) Genetic mutation. *Nature Education* 1(1):113**

1. Cualidad de las formas visuales que captan la atención del observador por la simplicidad, equilibrio o estabilidad de su estructura (R.A.E, 2006). [↑](#footnote-ref-1)