Extensión a UsiXML para el Soporte del Awareness

Jose Figueroa-Martinez [1], F. L. Gutiérrez Vela [1], Víctor López-Jaquero [2], Pascual González [2]

[1] Universidad de Granada
C/ Cuesta del Hospicio s/n - 18071
Granada, España
[2] Universidad de Castilla-La Mancha
CP 02071
Albacete, España
jfigueroa@ugr.es, fgutierr@ugr.es, victor@dsi.uclm.es, Pascual.Gonzalez@uclm.es

Abstract. (Draft) El soporte del Awareness en las tecnologías de desarrollo basadas en MDA es prácticamente inexistente. Esto se debe en gran parte a la falta de un modelo conceptual apto para caracterizar y representar el Awareness bajo las diferentes visiones existentes en las arquitecturas basadas en modelos. Este trabajo presenta una extensión al lenguaje de descripción de interfaces de usuario UsiXML, para dar soporte al Awareness como requisito de información. Este lenguaje tiene como objetivo describir interfaces de usuario multimodales y multicontexto. El enfoque basado en modelos de UsiXML lo hace un excelente candidato para integrar el soporte del Awareness desde la fase de ingeniería de requisitos hasta la implementación de las interfaces de usuario, permitiendo la trazabilidad de los requisitos de Awareness desde las interfaces de usuario finales a las tareas y entidades del dominio que los alimentan. Esto permitirá a los desarrolladores el mantener y validar todos los mecanismos de Awareness provistos por el sistema. El objetivo final del trabajo es obtener una mejora substancial en la calidad del software desarrollado, una clara mejora en los procesos de interacción y colaboración, un desarrollo organizado y trazable de los mecanismos de Awareness y un mantenimiento más sencillo.

Keywords: Awareness, model-based user interface development, requirements

1 Introducción

El desarrollo basado en modelos o MDD busca construir software a partir de un conjunto de modelos y procesos de transformación aplicados a ellos, en vez de codificar directamente el dominio, las tareas y las interfaces de usuario, como se hace en el desarrollo de software tradicional. El desarrollo de interfaces de usuario no es una excepción a la tendencia general a utilizar el diseño basado en modelos durante el desarrollo del software. Este interés en el desarrollo basado en modelos de interfaces

de usuario [1] ha sido expresado en la creación de un grupo en la W3C [2] para buscar la estandarización del diseño de interfaces de usuario basado en modelos. El enfoque dirigido por modelos aplicado al diseño de interfaces de usuario está soportado por un lenguaje de descripción de interfaces de usuario (UIDL) que provee un conjunto de elementos que dan significado a los modelos requeridos para describir el proceso de desarrollo de las interfaces de usuario. La revisión más reciente sobre los UIDLs disponibles puede encontrarse en [3]. Indudablemente, uno de los más activos UIDLs por sus herramientas disponibles y su comunidad es UsiXML [4].

UsiXML es un lenguaje de marcado basado en XML, creado para describir interfaces de usuario multi plataforma y multi contexto. Permite a los desarrolladores especificar interfaces de usuario a diferentes niveles de abstracción, separando los conceptos que afectan a los modelos usados en el desarrollo y a la propia aplicación, como es el dominio, las tareas, el contexto, los dispositivos etc. Dependiendo de estas especificaciones, las herramientas basadas en UsiXML pueden generar código especializado para cada plataforma objetivo.

UsiXML, así como otros UIDLs, permite describir un importante aspecto de desarrollo como es el contexto del usuario en donde las interfaces van a ser utilizadas. Pero dentro de este contexto no incluye el soporte a la especificación de requisitos de awareness o del proceso de awareness mismo, a pesar de que el awareness es primordial para la especificación y el funcionamiento de cualquier sistema colaborativo o de naturaleza dinámica.

De forma general, awareness significa "el conocimiento de lo que está pasando" [5]. Existen estudios extensos sobre el mismo y varios de ellos con muy diferentes interpretaciones de lo que se entiende por awareness. Para este trabajo, hemos seleccionado como base conceptual los modelos del awareness definidos por nosotros en [6], aunque los hemos madurado, completado y adaptado a las características de UsiXML Hemos realizado esta selección ya que estos modelos fueron desarrollados con el propósito de ser utilizados en metodologías de desarrollo basado en modelos, además de tener una robusta y estable base teórica: la teoría del "Situation Awareness" (SA) o Awareness de la Situación [5].

El objetivo de este trabajo es extender UsiXML paso a paso para introducir el Awareness como un nuevo requisito que pueda ser definido, relacionado, propagado y validado a través de toda la metodología.

2 Trabajos Relacionados. Formas de dar soporte al Awareness.

El soporte del Awareness ha sido mejorado ampliamente en herramientas y técnicas para el desarrollo tradicional (no basados en modelos). Desde toolkits gráficos como GroupKit [7] hasta marcos de trabajo para el soporte del Awareness, como el desarrollado por Drury [8]. Sin embargo, su soporte todavía está limitado principalmente a las últimas fases de este proceso, las cuales comúnmente son el diseño, la implementación y el mantenimiento del software.

El soporte del Awareness como un "añadido" en las últimas fases, representa un problema en el sentido de justificar su origen y su soporte de forma eficiente. Si toda

la implementación proviene de los requisitos (afirmación habitual bajo las metodologías de desarrollo de software modernas), los mecanismos de Awareness implementados también deben venir de los requisitos iniciales del problema.

El desarrollo basado en modelos está siendo cada vez más una sólida alternativa al desarrollo de software tradicional. Sin embargo, el soporte del Awareness es un componente que todavía falta en las tecnologías modernas de desarrollo basado en modelos. Esto se debe principalmente a la falta de un modelo del Awareness orientado al desarrollo, aunque en años recientes este tema ya ha sido retomado [6].

En el caso de UsiXML, Tesoriero [9] presenta una extensión al lenguaje para soportar las interfaces de usuario con awareness del usuario, es decir, interfaces especializadas para usuarios con distintas características. Sin embargo, la extensión propuesta por Tesoriero no contempla proporcionar ningún tipo de información de Awareness al usuario o técnicas especificas para modelarla. Aún así, hasta este momento, su propuesta es la única que contempla el Awareness como parte de UsiXML.

En relación con las interfaces de usuario y el proceso de interacción, los cuales son aspectos desafiantes del desarrollo de software y que tienen mucha relación con el soporte del Awareness, existen algunas herramientas que generan interfaces de usuario a partir de modelos de tareas. IdealXML [10] puede generar interfaces de usuario abstractas a partir del modelo de tareas. GrafiXML [11] puede generar interfaces de usuario concretas a través del uso de un editor gráfico.

A pesar de la importancia del Awareness y de su creciente adopción por las metodologías y herramientas de desarrollo de software, ninguno de los lenguajes de descripción de interfaces de usuario disponibles actualmente [12] soporta el Awareness como requisito inicial, aunque es verdad que lo toman en cuenta como elemento de importancia durante el diseño de las interfaces de usuario.

3 Introducción a UsiXML y al Awareness

UsiXML está basado en el Marco de Referencia Cameleon [12], el cual define los pasos del desarrollo de interfaces de usuario (UI) para aplicaciones interactivas multicontexto. En este proceso se identifican cuatro pasos: Modelado de Tareas & Conceptos (T&C), que representa las tareas a ser ejecutadas y los conceptos del dominio necesarios para llevar a cabo dichas tareas. Diseño de Interfaces de Usuario Abstractas (AUI), que define espacios de interacción a través de unir varias tareas de acuerdo a ciertos criterios (esto se hace a través de Objetos Abstractos de Interacción). Diseño de la Interfaz de Usuario Concreta (CUI), que representa una AUI a un nivel superior para un contexto específico de uso a través de Objetos Concretos de Interacción. Desarrollo de la Interfaz de Usuario Final (FUI), que representa la interfaz de usuario operativa. Estos pasos generan 4 capas de representación de la interfaz de usuario bajo diferentes visiones.

Los requisitos de Awareness deben ser representados en la capa T&C, ya que los aspectos de interacción y del dominio son definidos en esta capa. Analizando la metodología de definición de UsiXML, el proceso de abstracción/reificación y

transformación, el soporte del Awareness debe propagarse de la capa T&C hasta la capa de las FUI y ser trazable en las dos direcciones. Normalmente, el Awareness se usa en las interfaces de usuario como un proveedor de información, en donde dicha información es obtenida de las clases del dominio y del sistema en ejecución, como por ejemplo, la posición del cursor, la plataforma hardware del cliente, etc.

Pasando a la teoría, el Awareness es un estado de conocimiento. Endsley define el Awareness como "la percepción de los elementos en el entorno, en un volumen de tiempo y espacio, la comprensión de su significado y la proyección de su estado en el futuro cercano." Podemos tener Awareness de "algo", es decir, podemos tener un Awareness específico de un tipo caracterizado por "algo". El Awareness se usa para la toma de decisiones en entornos dinámicos y puede alcanzar niveles más altos de entendimiento de los elementos, así como proyecciones de sus valores en un futuro cercano.

Identificamos tres tipos de elementos de información para el observador: Elementos concretos, los cuales son recibidos directamente, Elementos compuestos, los cuales representan el entendimiento de otros elementos como un grupo (el sistema es capaz de interpretar varios valores concretos y según sus valores inferir un nuevo valor para todo el elemento), Proyecciones, las cuales representan los valores de uno o más elementos (concretos o compuestos) en el tiempo futuro calculados a través de mecanismos de inferencia (el sistema es capaz de inferir que valor tendrá el elemento cuando pase un tiempo determinado).

La información de Awareness generada por ordenador implica transmitir información (datos de Awareness) desde el sistema al usuario que los requiere. Esa información alimentará el estado de conocimiento del usuario que requiere Awareness cuando él se encuentra en el proceso de toma de decisiones y ejecución de tareas.

En este artículo, nuestra principal propuesta es el soportar Awareness proveyendo una forma precisa de representar requisitos relacionados con el Awareness, así como su integración con los modelos iniciales del UsiXML. Aplicando este enfoque a las metodologías basadas en modelos encontramos que el Awareness debe ser definido, relacionado con las tareas y transmitido a través de las interfaces de usuario, las cuales deben estar hechas para trasmitir datos de Awareness.

4 Soporte del Awareness en UsiXML

Para dar soporte al Awareness en UsiXML son necesarias algunas modificaciones iniciales: una forma de definir el Awareness, una forma de definir el uso del Awareness como requisito de información y una forma de enlazar y propagar los requisitos de Awareness a las FUI. A fin de conseguir esto, hemos desarrollado varios modelos y hemos añadido a los ya existentes, los cuales proveerán la estructura para el soporte del Awareness en UsiXML. El proceso de transformación estándar también debe cambiar a fin de generar los nuevos mapeos para permitir un soporte del "Awareness trazable".

Primero, presentaremos una forma de representar el Awareness según lo descrito por Endsley [5]. Después mostraremos como incluir el Awareness en un dominio

específico, a fin de obtener una fuente de datos de Awareness usable. Después de eso, describiremos como usar los datos de Awareness y como incluirlos en el proceso de desarrollo del software.

Para ejemplificar los requisitos de Awareness usaremos como caso de estudio un sistema para el manejo de camiones de bomberos ("Fire truck" o autobomba) para una Central de Bomberos. El sistema puede ser manejado por un bombero experto y por uno auxiliar. Cada uno tiene diferentes conocimientos, pero los dos tienen la misma tarea: enviar las autobombas requeridas para cubrir las emergencias del momento. La entidad para representar la autobomba tiene los siguientes atributos: `id`, `name`, `gas capacity`, `water capacity`, `location`, `onmaintenance`.

El bombero experto requiere toda la información actualizada de las autobombas. El bombero auxiliar requiere la misma información pero previamente "asimilada" por el sistema, como es el caso del elemento `availability`. Además requiere una proyección del mismo elemento para conocer su estado en un futuro cercano.

Representación genérica del Awareness

Siguiendo la descripción del Awareness hecha por Endsley, definimos el Awareness de "algo" (un recurso, una localización, sus cambios de estado, etc.) a través de un conjunto de elementos de información. Los "Elementos" representan algo que el observador puede recibir y entender, como por ejemplo, la localización, peso, altura, velocidad, etc. Estos elementos concretos son características de la entidad observada.

Puede haber otros elementos creados por el entendimiento del propio observador, los cuales llamamos *elementos compuestos*. Definimos los elementos compuestos como una composición de elementos concretos y/o elementos compuestos a través de una función de composición. También puede haber proyecciones de los valores de algunos elementos en el futuro. Las llamamos *proyecciones de Awareness*. Fig. 1 a.

Estos elementos no poseen una medida o tipo de dato específico, como metros, kilogramos, u otros. Son elementos abstractos y por ello forman un tipo de Awareness abstracto. Entonces, un Tipo Abstracto de Awareness (AAT) es una representación de una estructura abstracta de elementos que forma un Tipo de Awareness.

El modelo del AAT puede ser útil para compartir y reutilizar modelos genéricos de AATs comunes como son el Group Awareness, Presence Awareness, Location Awareness, Workspace Awareness, entre otros. Sin embargo, no los describimos en este trabajo porque explícitamente hablando, no son requeridos como tal para el soporte del Awareness aunque sí consideramos que su definición precisa permitiría tener un catalogo de awareness que facilitaría el modelado de sistemas complejos.

Para nuestro ejemplo, crearemos el *Fire truck Awareness* y el *Location Awareness*. El *Fire truck Awareness* tiene los siguientes elementos: `gas`, `water`, `id`. Además, tiene un elemento compuesto: `availability`, el cual depende de `gas` y `water`. El *Location Awareness* tiene los siguientes elementos: `location`, `id`.

Estos tipos de Awareness son AATs y por ahora solamente especifican una estructura abstracta de datos de Awareness que no está ligada a ningún dominio concreto.

Para definir un Tipo de Awareness especializado a un dominio utilizaremos el siguiente modelo, el cual posee la misma estructura conceptual que el AAT, pero mapeada a atributos de las clases del dominio.

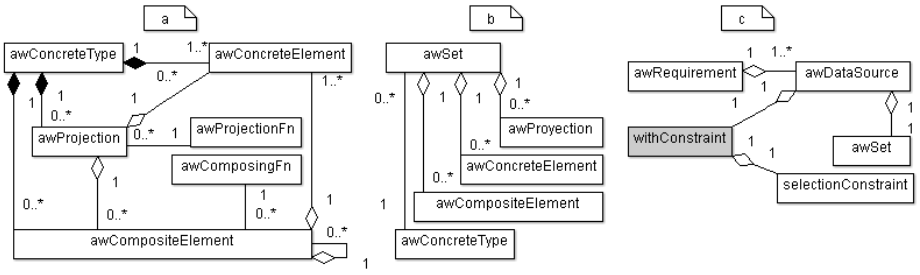

Fig. 1. a) Modelo del Tipo de Awareness Concreto, el cual está compuesto por elementos concretos y compuestos de Awareness y posibles proyecciones de los mismos. b) Awareness Set. Permite definir un conjunto de elementos y/o proyecciones de Awareness para usarse como estructura de datos proporcionados a un usuario. c) Requerimiento de Awareness y Fuente de Datos de Awareness. Permiten definir un conjunto de datos de Awareness accesible por las tareas que lo utilicen.

Representación del Awareness específica a un dominio

Se requiere una forma de definir tipos de Awareness para las entidades de un dominio específico. Estos tipos de Awareness deben ligarse a los atributos de las clases del dominio a fin de ser usables como fuentes de datos de Awareness. Los denominamos Tipos Concretos de Awareness (ACT) y representan la forma de definir el Awareness soportado en un sistema. En diferentes dominios debemos poder definir muchos ACTs, algunos de ellos del mismo AAT. Eso no debe ser una restricción.

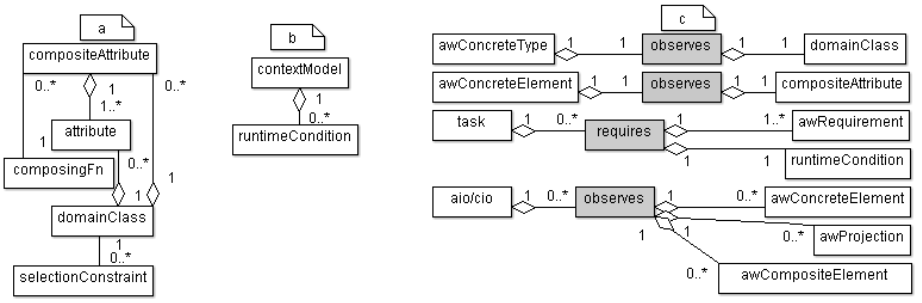

Fig. 2. a) Atributo Compuesto. Permite agrupar atributos de clases del dominio a través de una función de composición. Necesario para definir un Elemento de Awareness. b) Condición durante tiempo de ejecución. Permite describir una condición o predicado a comprobarse en tiempo de ejecución y usable para condicionar el uso de mecanismos o interfaces de usuario. c) Entidades para enlazar los modelos del Awareness con tareas e interfaces de usuario.

Para definir un ACT se crea una estructura (similar a la estructura del AAT base) de elementos de Awareness ligada con la entidad fuente de datos y con los elementos de Awareness ligados a los correspondientes atributos (*compositeAttribute*) de la clase

observada, como se muestra en la Fig. 2 c. Proponemos extender el Modelo del Dominio con las entidades mostradas en la Fig. 2 a.

En nuestro ejemplo, el *Firetruck Awareness* y el *Location Awareness* de la entidad *Firetruck* es requerido por los usuarios. A continuación definimos los ACTs correspondientes enlazando los elementos de Awareness con los *compositeAttribute* que observan:

`myFiretruckAw [gas -> cgas_capacity, water -> cwater_capacity, id -> cid, availability (gas, water), pavailability (availability)], myFtLocationAw [location -> clocation]`.

Nótese que los atributos con una "c" al principio identifican attributos compuestos y los elementos que comienzan con "p", como `pavailability`, identifican proyecciones de otro elemento, la cual sirve para conocer el valor estimado de ese elemento en el tiempo futuro.

Representación de los requisitos de Awareness

Ahora que podemos definir tipos de Awareness para un dominio específico, el siguiente paso es definir su uso. Para este propósito proponemos cambios en algunos modelos de UsiXML a fin de tratar varias dificultades inherentes al soporte del Awareness como son la privacidad y la distracción.

La privacidad implica que los datos de Awareness solo deben proporcionarse a los usuarios que los necesitan y solo cuando lo necesiten y en donde los necesiten. Para definir este contexto, proponemos añadir la entidad *Runtime Condition* (RC) al Modelo del Contexto. Ver Fig. 2 b. La distracción es causada por dar al usuario demasiados o muy pocos datos de Awareness (además de otros factores propios del usuario, como la concentración, el enfoque, el estado de ánimo, etc.). Damos una solución a la distracción permitiendo definir explícitamente los datos de Awareness que se le deben proporcionar al usuario en cada situación utilizando para ello las entidades *awSet* y *awDataSource*. Fig. 1 b & c.

Siguiendo la teoría del Awareness, el proceso de toma de decisiones se produce durante la ejecución de las tareas de un sistema. Toda la información y los datos requeridos durante una tarea deben proporcionarse al usuario que lo requiera. Esto significa que durante una tarea el usuario o los usuarios pueden recibir los datos actualizados que requieren y que pueden ser distintos dependiendo de la situación.

Nuestra propuesta es agregar (a través de mapeos [14]) requisitos de Awareness (AR) a las tareas. Pero, ¿qué es un AR? Es la representación de un requisito de información de Awareness (Fig. 1 c) durante un caso o contexto específico (RC) accediendo a un grupo selecto de instancias de la fuente de datos de Awareness para obtener esta información. Fig. 2 c.

Una tarea puede albergar varios ARs dependiendo de los distintos RCs ligados a ellos. Esto se traduce en que para cada situación (RC) la tarea ofrecerá al usuario un selecto grupo de datos de awareness, según estén especificados en los *awDataSource* (e implícitamente en los *awSet*) a través de distintas interfaces de usuario similares, pero distintas, dependiendo de los datos que transmitan.

Para representar los ARs de nuestro ejemplo primero debemos definir los grupos de elementos y fuentes de datos de Awareness que van a utilizarse, usando los modelos presentados en la Fig. 1 b & c. Para definir una fuente de datos de Awareness se utiliza también la entidad *selectionConstraint*, la cual permite definir una restricción de selección para elegir un grupo de instancias de la fuente de datos ligada a un *awSet* base para el *awDataSource*. Un *awRequirement* es un conjunto de *awDataSource*s, el cual se enlaza con una tarea a través del mapeo llamado *requires*, el cual permite agregarle a una tarea uno o varios *awRequirement* en la situación o contexto definido por una *runtimeCondition* (Fig. 2 c), la cual puede expresarse en un lenguaje como OCL, SQL, entre otros.

Para nuestro ejemplo, tenemos la tarea `ftManagementTask` la cual va a albergar un AR pero relacionado con condiciones distintas del usuario en ese momento: cuando el usuario es experto (`rcExpert`), y cuando el usuario es auxiliar (`rcAux`).

El bombero experto requiere todos los elementos concretos del `myFiretruckAw` y del `ftLocationAw`. Esto se representa mediante la instancia `ftExpertSet` [`gas, water, id`] y la instancia `ftLocationSet` [`id, location`].

El usuario auxiliar requiere el conocimiento a nivel de comprensión [5], el cual se representa mediante elementos compuestos del Awareness, como es el elemento `availability`, además de una proyección del mismo elemento. Esto se representa con la instancia `ftAuxSet` [`id, availability, pavailability`]. Con estos tres *awSets* definidos podemos especificar las fuentes de datos de Awareness: `ftExpertSource` {source: `ftExpertSet`, withConstraint: `slExpertUser`}, `ftAuxSource` {source: `ftAuxSet`, withConstraint: `slAuxUser`}, `ftLocationSource` {source: `ftLocationSet`, withConstraint: `slAllUsers`}.

Las instancias que inician como "`sl`" representan *selectionConstraints*.

El siguiente paso es definir los dos ARs identificados, los cuales se componen de varias fuentes de datos de Awareness: `ftExpertReq` [`ftExpertSource, ftLocationSource`], `ftAuxReq` [`ftAuxSource, ftLocationSource`].

Una vez definidos los Requerimientos de Awareness pueden ligarse a la tarea a través del proceso de mapeo: `ftManagementTask` [requires: {req: `ftExpertReq`, rc: `rcExpertUser`}, requires: {req: `ftAuxReq`, rc: `rcAuxUser`}].

La entidad de mapeo *requires* es la encargada de relacionar tareas y Requerimientos de Awareness. Una vez establecidas estas relaciones, el soporte básico del Awareness está definido. Aquí un ejemplo de cómo se verían estos requerimientos de Awareness en UsiXML:

```
<mappingModel id="mmDomainFiretruck" name="mmDF">
 <!-- Previamente definidos otros mapas -->
 <requires id="reqManagmentTaskExpert">
   <source sourceid="ftManagementTask"/>
   <target targetid="ftExpertReq"/>
   <target targetid="rcExpertUser"/>
 </requires>
```

```
  <requires id="reqManagementTaskAux">
    <source sourceid="ftManagementTask"/>
    <target targetid="ftAuxReq"/>
    <target targetid="rcAuxUser"/>
  </requires>
</mappingModel>
```

5 Conclusiones y Trabajo Futuro

Las técnicas de desarrollo basadas en modelos abordan importantes problemas del proceso del desarrollo de software. Tienen un riguroso manejo de requisitos y puntualizan su justificación y cumplimiento. Han sido extendidas para manejar las interfaces de usuario de la misma forma, permitiendo llevar los requisitos del sistema y su utilización hasta las interfaces finales de usuario y de esta forma hacer la vida más fácil a desarrolladores y usuarios de estos sistemas complejos e interactivos.

Este trabajo presenta un enfoque basado en modelos para el desarrollo de interfaces de usuario con soporte de Awareness basándose en la metodología UsiXML El nuevo aspecto en este enfoque es que el Awareness se define como un conjunto de requisitos iniciales y es llevado a través de todos los estados de la metodología, permitiendo requisitos de Awareness trazables, verificables y reusables.

La separación lógica entre el Awareness como requisito y las interfaces de usuario mejora la calidad y mantenibilidad de las dos parte, además crea nuevas maneras de mejorar el proceso de desarrollo.

El soporte del Awareness definido como otro modelo abre la puerta a otras formas de abstracción y genera ventajas relacionadas con el tratamiento de los modelos a través del proceso de desarrollo y ejecución final del software. El conocimiento almacenado en los modelos de Awareness puede ser usado en novedosas formas y generar más ventajas para los desarrolladores y usuarios finales. Los mecanismos de adaptación también pueden mejorar a través de estos modelos.

Algunos problemas importantes relacionados con el soporte del Awareness como la privacidad y la distracción son abordados a través del uso de "runtime conditions" o "condiciones en tiempo de ejecución" junto con los requisitos de Awareness, y a través del uso de "selection constraints" o "restricciones de selección" para la selección de las fuentes de datos de Awareness.

Los modelos del Awareness no desechan otras herramientas o técnicas para el soporte del Awareness. Estos modelos pueden verse como de "bajo nivel" para representar los requerimientos de Awareness en el sistema.

Como trabajo futuro, estamos trabajando en mejorar la integración de los modelos del Awareness con la estructura y los procesos de transformación de UsiXML. Además, planeamos incluir estos modelos en otros lenguajes de definición de interfaces de usuario así como en otras metodologías basadas en modelos.

Agradecimientos

Esta investigación ha sido financiada por el Ministerio de Ciencia e Innovación de España como parte del Proyecto DESACO (TIN2008-06596) y por el Concejo Nacional de Ciencia y Tecnología de México CONACYT.

Referencias

1. Puerta, A.: A Model-Based Interface Development. In: IEEE Software, vol. 14, No 4, pp. 40—47. (1997)
2. Cantera, J.M.: Model-Based UI XG Final Report. W3C Incubator Group Report 04 May 2010, http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui/ (2010)
3. Guerrero-García, J., González-Calleros, J.M., Vanderdonckt, J., Muñoz-Arteaga, J.: A Theoretical Survey of User Interface Description Languages: Preliminary Results. In: Proc. of LA-Web/CLIHC'2009 (Mérida, Nov 9-11, 2009), pp.36-43. IEEE Computer Society Press, Los Alamitos (2009)
4. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López Jaquero, V.: UsiXML: a Language Supporting Multi-Path Development of User Interfaces. In: Proc. EHCI-DSVIS'2004. LNCS, vol. 3425, pp. 200-220. Springer-Verlag, Berlin, Germany (2005)
5. Endsley, M.R.: Towards a Theory of Situation Awareness in Dynamic Systems. In: Human Factors, vol. 37, issue 1, pp. 32—64, (1995)
6. Figueroa Martinez J., Gutierrez Vela F. L., Collazos C. A.: Awareness Models for the Development of Ubiquitous Systems. In Juan Carlos Augusto, Juan M. Corchado, Paulo Novais, and Cesar Analide (Eds.), ISAmI, (72):237-245, Springer (2010)
7. Roseman M., Greenberg S.: Building Real-Time Groupware with GroupKit, A Groupware Toolkit. In: TOCHI, vol. 3, issue 1, pp. 66—106 (1996)
8. Drury, J., Williams, M.G.: A Framework for Role-Based Specification of Awareness Support in Synchronous Collaborative Systems. In: Proc. of the 11th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02) 1080--1383/02. Pittsburgh, PA (2002)
9. Tesoriero R., Vanderdonckt J.: Extending UsiXML to Support User-Aware Interfaces. In: R. Bernhaupt et al. (Eds.) HCSE 2010. LNCS, vol. 6409, pp. 95--100. (2010)
10. Montero, F., López Jaquero, V.: IdealXML: An Interaction Design Tool and a Task-Based Approach to User Interface Design. In: 6th International Conference on Computer-Aided Design of User Interfaces (CADUI 2006), Bucharest, Romania, pp. 6—8 (2006)
11. Michotte, B., Vanderdonckt, J.: GrafiXML, a Multi-target User Interface Builder Based on UsiXML. In: Proc. ICAS2008, Los Alamitos, USA. IEEE Computer Society Press (2008)
12. Proc. of the 1st Int. Workshop on USer Interface eXtensible Markup Language. (UsiXML '2010) ACM SIGCHI Symposium on Engineering Interactive Computing Systems (2010)
13. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. In: Interacting with Computers 15,3 (June 2003), pp. 289—308. Elsevier (2003)
14. Montero, F., López Jaquero, V., Vanderdonckt, J., González, P., Lozano, M.D., Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML. DSV-IS'2005), Newcastle upon Tyne, England, July 13-15, 2005. LNCS 3941, Springer-Verlag, Berlin, Germany, ISSN: 0302-9743, (2005)