REVISTA Universidad EAFIT
Vol. 41. No. 138. 2005. pp. 44-64

Modelo de contexto
para realidad aumentada

Andrés Agudelo Toro
Ingeniero de Sistemas Universidad EAFIT.
Asistente de Investigación del Grupo de Investigación en Sistemas de Control Digital del Departamento de Ciencias Básicas de la Universidad EAFIT.
aagudel6@eafit.edu.co

Recepción: 19 de enero de 2005 | Aprobación: 10 de mayo de 2005

Resumen

Las tecnologías de Realidad Aumentada (RA) permiten presentar información virtual en el mundo real. Lograr la combinación real-virtual en RA es complejo y no consiste simplemente en sobreponer un mundo sobre el otro. Las aplicaciones de RA requerirán de métodos de abstracción apropiados para estar al tanto de lo que sucede en el contexto. En este trabajo se presenta un modelo de datos orientado a objetos del contexto para RA. Las ventajas encontradas en el uso de este modelo de datos se comparan con el modelo tradicional. Se concluye que el modelo tradicional es limitado y debe hacerse uso del modelo de contexto.

Palabras Clave

Realidad Aumentada
Modelos de Datos
Diseño de Software
Computación Ubicua
Aplicaciones dependientes del contexto
Patrones de Diseño

Context model for Augmented Reality

Abstract

Augmented Reality (AR) technologies allow to present virtual information in the real world. An Augmented Reality system differs from the traditional systems because it interacts to a great extent with the user and his surroundings. The virtual-real combination in AR is complex and it does not consist simply on superposing the two worlds. AR applications require appropriate methods of abstraction to be aware of the environment or context. This paper presents an object oriented context model for AR. In this model, context information is represented as a set of objects, describing real or virtual entities, structured in a graph. The advantages found in the use of this data model are compared with a traditional model. The conclusion is that the traditional model is limited and context models should be used.

Key Words

Augmented reality
Data models
Software design
Ubicuous computing
Context dependant applications
Design patterns

Introducción

La línea entre el mundo real y el mundo virtual es cada vez más delgada. La información de los sistemas digitales es cada vez más parte de las actividades diarias de las personas. La Realidad Aumentada (RA) se encuentra en la línea entre el mundo real y la información del mundo digital. La RA permite presentar información digital en el mundo real por medio de dispositivos de representación especiales (Azuma, 1995). Una aplicación de Realidad Aumentada se separa de los sistemas tradicionales, pues la interacción con el usuario y el entorno es más alta y a mayor velocidad. Desarrollar sistemas de RA representa retos en cuanto a métodos de procesamiento y representación de información. No consiste simplemente en sobreponer geométricamente un mundo sobre el otro. El salto de la información digital al mundo requiere de metodologías apropiadas para hacer que los sistemas conciban lo que está sucediendo en el ambiente y puedan reaccionar ante los eventos que allí ocurren. Los métodos de abstracción son de vital importancia. Los modelos de datos son claves en tiempo de desarrollo y ejecución. Las aplicaciones de Realidad Aumentada actuales se ven limitadas por el poco conocimiento que tienen del mundo. Su dependencia en los métodos tradicionales de desarrollo de sistemas de gráficos tridimensionales y de Realidad Virtual ha restringido su acceso a la riqueza de información del entorno. Los modelos de datos heredados de tales aplicaciones fueron efectivos dentro de su alcance, pero son limitados para los requerimientos de la RA.

Se ha dado por hecho que las aplicaciones que dependen en gran medida de la información del mundo real requieren de nuevas metodologías de diseño (Norman, 1998; Banavar *et al,* 2000; Henricksen *et al*, 2001; Satyanarayanan, 2001). En específico, este trabajo considerará el modelo de datos para sistemas RA. Un modelo de datos para sistemas de computo se describe como un formalismo matemático, que consta de una notación para describir la representación de los datos en un sistema de software y de un conjunto de operaciones válidas que se pueden usar para manipularlos (Graham, 1991). En una definición más práctica, un modelo de datos es la abstracción usada para describir un problema en un sistema de software (Aho y Ullman, 1995). Una aplicación de RA requiere de un modelo de datos apropiado para manejar la información del mundo real, que sea flexible, extensible y con la capacidad para representar información del mundo real y virtual. Modelos más apropiados impactarán positivamente el tiempo de desarrollo y harán los sistemas de RA más efectivos, prestando un

mayor beneficio al usuario. Otra área de investigación, aparte de la RA, ha reconocido la existencia del problema de la representación del mundo. La ***Computación Ubicua*** (Weiser,1995) y las ***Aplicaciones Dependientes del Contexto*** (Schilit *et al*, 1994), estudian los sistemas de software que actúan de acuerdo con la información que tienen del lugar donde se ejecutan, también llamado contexto. Investigaciones en esta área han permitido establecer las características principales del contexto y ha provisto soluciones cercanas, más no completas, de representación del mundo.

Este trabajo busca proponer un modelo de contexto propio para aplicaciones de Realidad Aumentada. Las aplicaciones de Realidad Aumentada requieren de un modelo de datos diferente al modelo de datos de las aplicaciones tradicionales de graficas tridimensionales y de Realidad Virtual. Se justifica la existencia de tal necesidad y se presentará un modelo que satisface un conjunto de requerimientos establecidos de acuerdo con un análisis del modelo actual. Se presenta un marco de referencia en Realidad Aumentada y Computación Ubicua, las deficiencias de las arquitecturas tradicionales y una lista de requerimientos del modelo buscado. Para definir el modelo se toma un caso de aplicación que refleja los requerimientos y luego se describe el modelo de contexto. Para demostrar su aplicabilidad se implementa y describe el sistema propuesto basándose en el modelo y se evalúa el modelo. El modelo propuesto se define como un patrón de diseño. Al final se obtienen conclusiones y se propone como puede continuar esta investigación en un trabajo futuro.

1. Trabajos Relacionados

Este trabajo de fundamenta en la visión de la Computación Ubicua y las Aplicaciones dependientes del Contexto de Weiser (1995) y Schilit et al (1994), la definición de contexto de Dey y Abowd (2000), la descripción de modelos de datos y sistemas de computo de Aho y Ullman (1995); la definición de Realidad Aumentada de Azuma (1995); el estudio de arquitecturas de RA de Brügge *et al* (2002); se toman los principios del modelo de contexto de Henricksen *et al* (2002) y la formalización de Gyssens et al (1994).

Las estructuras y modelos de datos no han sido de especial preocupación para el área de Realidad Aumentada. Se presta más atención a métodos de ubicación (Kato y Billinghurst, 1999; Azuma *et al,* 2001) y

arquitectura general del sistema (*Schmalstieg* et al, 2002; Reicher *et al*, 2003; Reitmayr, 2004). El uso del *Scene Graph* como modelo de datos central es generalizado, pues siempre se ha partido de la realidad aumentada como una extensión de la Realidad Virtual (Azuma, 1995; Brügge *et al*, 2002). Estudios arquitectónicos en general han reconocido la necesidad de mantener la información del contexto (Brügge *et al*, 2002) pero no se han buscado modelos para representarlo. Newman *et al* (2004) considera el uso de un modelo de datos basado en grafos para el sistema de ubicación, sin embargo se usa exclusivamente para coordinar los sensores del sistema. Se han presentado modelos de represtación del contexto en Kindberg *et al* (2000), Harter *et al* (1999) y Schmidt *et al* (1999). Tales modelos son limitados a aplicaciones específicas, en especial de computación móvil, y a tipos de contexto restringidos, ninguno de RA. Henricksen *et al* (2001) presenta un modelo general para aplicaciones dependientes de contexto, sin embargo tal modelo maneja un conjunto limitado de relaciones entre los elementos del contexto. Este trabajo presenta un modelo de contexto con un enfoque en sistemas RA.

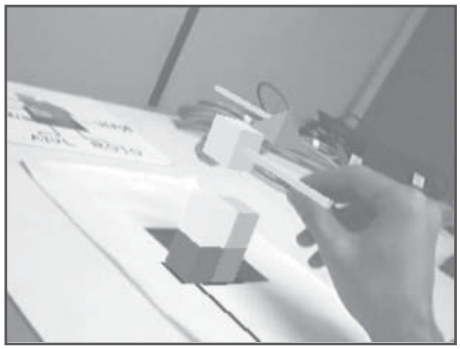
2. Realidad Aumentada

La Realidad Aumentada (RA) es la mezcla de información computacional con el mundo real. En una definición clásica, la realidad aumentada es un tipo de ambiente virtual en el cual el usuario no se sumerge completamente en un mundo virtual sino en una mezcla de éste con el mundo real. Para el usuario aparecen los objetos virtuales y reales coexistiendo en el mismo espacio. La Figura 1 muestra un ejemplo de realidad aumentada en el que cubos virtuales pueden ser observados en la realidad por un usuario.

Un sistema de realidad aumentada cuenta con las siguientes características:

- La información digital es combinada con la realidad.
- La combinación de lo real y lo virtual se hace en tiempo real.

Figura 1. Cubos virtuales aumentando la realidad



- La información aumentada se localiza o "registra" en el espacio. Para conservar ilusión de ubicación real y virtual, esta tiende a conservar su ubicación o moverse respecto a un punto de referencia en el mundo real.

En cuanto a su funcionamiento, las aplicaciones de RA tienen tres subsistemas fundamentales: visualización (salida), ubicación de objetos virtuales en el mundo real (registro) y métodos de interacción (entrada):

Visualización. Se logra con el uso de dispositivos de visualización similares a los de Realidad Virtual. Algunos de estos dispositivos son cascos y gafas. Éstos se componen por pantallas de cristal líquido funcionando como si fueran lentes trasparentes para que pueda observarse el mundo real y permitir adicionar los objetos virtuales. En la Figura 2 se observa la combinación real-virtual y dos métodos para llevarla a cabo.

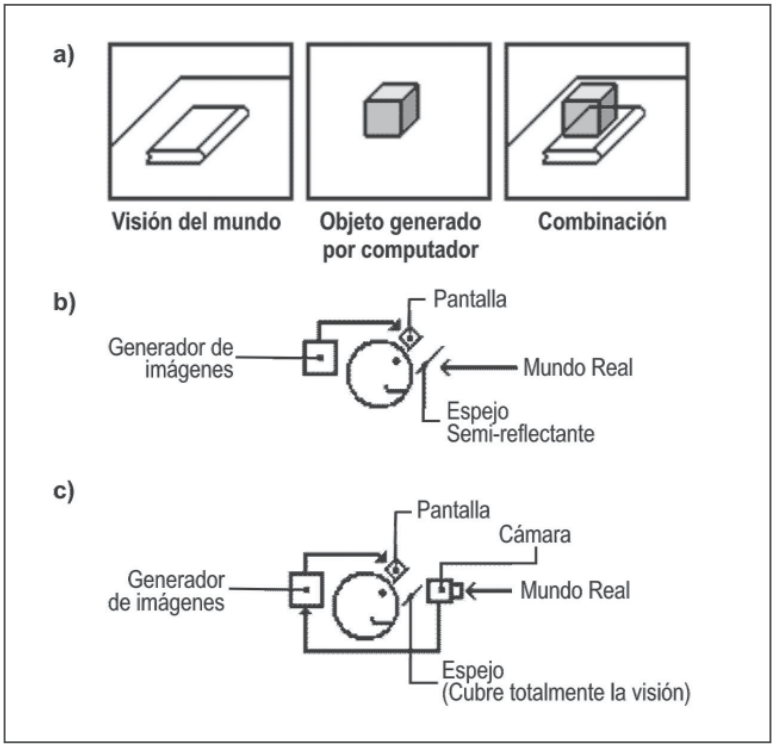
Registro de objetos virtuales. Consiste en lograr que los objetos virtuales puedan "registrarse" con el mundo real, de tal forma que cuando el usuario se mueva los objetos parezcan conservar su posición.

Interacción. Consiste en métodos para manipular o modificar tales objetos.

2.1 Experiencias de desarrollo en Realidad Aumentada

Los sistemas de RA requieren métodos de desarrollo especiales, incluyendo técnicas avanzadas de programación orientada a objetos,

Figura 2. Métodos: a) Combinación de visión con objetos virtuales, b) Método directo, c) Método Indirecto



sistemas distribuidos e interfaces gráficas. En particular, la representación del mundo y la forma de combinar los elementos del mundo real y virtual son de especial atención. El desarrollo de dos sistemas de RA permitió conocer exigencias comunes y encontrar carencias de los métodos de desarrollo actuales: ARMotodo y *Parallel Worlds Framework*.

Durante la construcción de ARMotodo[1] (ARM) se experimentó con sistemas que combinaban realidad aumentada y realidad virtual. ARM es un juego de bloques de construcción que permite que dos usuarios ubicados, uno en un ambiente virtual y otro en el real, puedan fabricar conjuntamente una figura tridimensional compuesta de bloques. Los bloques de ARM son completamente virtuales. Éstos pueden ser manipulados por el usuario en el mundo real por medio de un dispositivo de rastreo que funciona a manera de un ratón de computadora tridimensional. El usuario en el mundo virtual (para este caso un computador de escritorio), tiene una visión simulada del mundo real y manipula los bloques con el ratón.

Para ARM fue necesario contar con un modelo de datos unificado. Los elementos principales, los bloques, son representaciones visibles de entidades virtuales, con características principalmente de forma y ubicación en el espacio. Se seleccionó para el sistema un modelo de datos basado en una matriz tridimensional, donde cada bloque era representado por una estructura de datos que podía ocupar una de las posiciones en la matriz. Este modelo de datos era simple y representaba bien la situación que venía desarrollándose de elaboración de figuras. Aparte que permitía implementar fácilmente detección de colisiones y simular el efecto de la fuerza de gravedad. El estado del modelo debía ser actualizado en tiempo real por múltiples instancias de la aplicación en

[1] El sistema ARMotodo se construyó en 2002 y se encuentra disponible ante petición.

diferentes máquinas, a manera de los sistemas de realidad virtual distribuida. Se trató de un diseño simple basado en una estructura de datos estática y de mínima escalabilidad, propio de una aplicación de escritorio.

El sistema de colaboración a distancia *Parallel Worlds* fue el segundo paso en la evolución de un sistema de colaboración a distancia entre usuarios de realidad virtual y aumentada (Ganapathy *et al*, 2003). El *Parallel Worlds Framework* (PWF) integraba un sistema de colaboración distribuida, uno de realidad virtual y uno de realidad aumentada. PWF agregaba varios niveles de complejidad al sistema de colaboración a distancia original. PWF debía soportar múltiples usuarios en el mundo real y virtual. Otro tipo de usuario, un agente inteligente, hacia parte de la colaboración. La interacción era más compleja, pues tanto objetos virtuales como reales debían ser manipulados por el usuario. Otras formas de comunicación con el sistema, como reconocimiento de voz, debían incluirse. El sistema final estaba formado por un conjunto de sub-aplicaciones con tareas específicas de presentación en realidad virtual o presentación en realidad aumentada. Las sub-aplicaciones necesitaban conocer hasta cierto punto el estado general de la situación del mundo real-virtual. En específico se requería conocimiento de los objetos virtuales y reales, los usuarios, los dispositivos y las relaciones entre los anteriores.

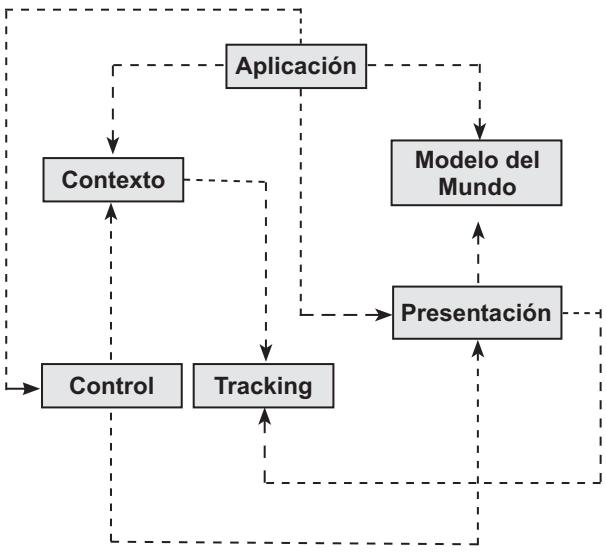
La solución propuesta para incluir la información de objetos virtuales y reales, usuarios, dispositivos y relaciones entre estos fue el *Scene Graph* (SG), pues el sistema distribuido estaba basado en el *middleware* de DISCIPLE (Marsic, 1999), cuyo un modelo de datos es un SG replicado. Debido a esto se buscó una forma de organizar jerárquicamente los elementos, pero se presentaron limitaciones. El modelo jerárquico no permitía expresar relaciones distintas a las espaciales. La jerarquía diseñada era además compleja y no representaba bien la situación del entorno que se pretendía expresar. Aparte de esto el SG tuvo que extenderse para agregar atributos adicionales a los elementos y debió adicionarse representaciones virtuales de los objetos reales para poder manejarlos. En la implementación final la información de la situación quedó dispersa en el sistema y se vio la necesidad de agregar un protocolo adicional de paso de mensajes para mantener el estado.

La experiencia adquirida en el desarrollo de ARM y PWF y el conocimiento de otras aplicaciones de la misma naturaleza y sus deficiencias como se presenta en 2.2, llevó a la búsqueda un modelo más adecuado para representar la situación del mundo. Este modelo se presentará en la sección 4.

2.2 Sistemas actuales de Realidad Aumentada

Brügge *et al* (2002) presenta un resumen de las arquitecturas de los sistemas de software de realidad aumentada más representativos. El estudio incluyó 18 arquitecturas y de allí se extrajo una arquitectura de referencia que contiene los componentes comunes de tales sistemas. El diagrama de componentes y dependencias puede observarse en la Figura 3. La función de cada componente se define a continuación:

Figura 3. Arquitectura de referencia



Fuente: Brügge, Bernd, MacWilliams, Asa y Reicher, Thomas (2002). "Software architectures for augmented reality systems – report to the ARVIKA consortium". En: *Technical Report.* Technische Universität München. TUM-I0410.

- **Aplicación:** Maneja la lógica y contenidos específicos del sistema.
- **Tracking:** Determina la posición de usuarios y objetos.
- **Control:** Procesa las entradas para el usuario.
- **Presentación:** Se encarga de la representación gráfica.
- **Contexto:** recoge diferentes datos de contexto.
- **Modelo del Mundo:** almacena información sobre los objetos virtuales y reales.

Estas arquitecturas no consideran la información del contexto y el modelo del mundo como un todo. No se habla de un ***modelo del contexto.*** El componente contexto sólo se encarga de procesar la información del subsistema *Tracking* y repartirla a otros subsistemas. El modelo del mundo sólo incluye los objetos reales o con representación gráfica, algunos sistemas sólo incluyen los objetos virtuales. Según el estudio, el modelo de mundo es el *Scene Graph*, que es limitado a relaciones geométricas. El usuario como tal no se considera elemento de ningún componente, a pesar de la importancia que tiene en las aplicaciones dependientes del contexto (Schilit *et al* ,1994; Dey y Abowd, 2000; Henricksen *et al*, 2001).

Agregando lo anterior a la experiencia adquirida en el desarrollo en realidad aumentada permite, concretar las consecuencias de estas fallas en las arquitecturas de realidad aumentada. Las fallas se reflejan en dos aspectos fundamentales:

- Desconocimiento de la aplicación de la situación del mundo real. Los elementos adicionales del contexto no están en el modelo. Lo que se conoce del mundo (sólo los objetos) se describe únicamente como relaciones espaciales geométricas. En la mayoría de los casos éstas no son suficientes.
- Dificultad en la implementación del sistema de RA. No tener acceso directo a la situación del contexto hace complicada la implementación de la lógica de la aplicación. El acceso al contexto, se hace mediante operaciones indirectas en distintos subsistemas.

2.3 Exigencias del nuevo modelo de datos

Para crear aplicaciones dependientes del contexto no se puede partir simplemente de extender los métodos de diseño procedentes de las aplicaciones de escritorio. La Realidad Aumentada tomó su modelo principal de la Realidad Virtual y esta a su vez de las aplicaciones gráficas. Se define una lista de requerimientos que debe satisfacer el nuevo modelo. Un modelo de contexto para realidad aumentada debe cumplir por lo menos con cinco requerimientos:

- El modelo debe permitir contener información sobre todos los elementos del contexto (mundo real) y del sistema (mundo virtual) relevantes a la aplicación.
- El modelo debe ser unificado. La información no debe estar dispersa por el sistema. La forma de acceder a la información debe ser genérica.
- El modelo debe ser escalable y extensible. Debe soportar múltiples tipos de aplicaciones y situaciones de contexto. Por ejemplo, no debe permitir solamente manejar la información de un usuario, sino de un creciente número de éstos.
- El modelo debe abstraer la situación de manera tal que sea comprensible para el diseñador y el sistema.
- El modelo debe poder expresarse formalmente y ser válido dentro de la teoría de modelos de datos.

Se presentarán algunos aspectos necesarios sobre la computación ubicua antes de definir el modelo propuesto.

3. Computación ubicua y aplicaciones dependientes del contexto

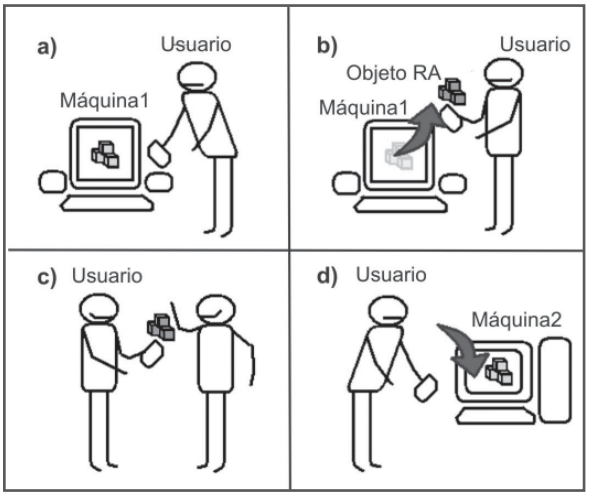
Las tecnologías de cómputo son ubicuas en la medida que se encuentran en más lugares y hacen cada vez más parte de nuestra vida cotidiana (Weiser,1995). Las aplicaciones que permiten la computación ubicua pueden clasificarse como dependientes del contexto en que se encuentra el usuario. Al proveer acceso al contexto[2] se incrementa la riqueza de la comunicación hombre-máquina y la efectividad de la elaboración de la tarea. Una ***aplicación dependiente del contexto*** se definió inicialmente como un ***sistema que examina y reacciona ante el contexto en que se encuentra el usuario*** (Schilit *et al,* 1994). Hull *et al* (1997) las define como los ***sistemas de cómputo capaces de sensar, interpretar y responder de acuerdo con el entorno en que se encuentra el usuario.*** Para Dey y Abowd (2000) un sistema es dependiente del contexto si hace uso del contexto para proveer información o servicios relevantes al usuario, de acuerdo con la tarea que éste se encuentre desarrollando. De acuerdo con Henricksen *et al* (2001) la información de contexto exhibe distintas características temporales, es imperfecta, tiene diferentes representaciones y está altamente relacionada.

3.1 Representación del contexto

El contexto es una fuente rica en información, que requiere modelos de representación avanzados. El contexto puede llegar a tener múltiples representaciones alternativas. El punto clave está en encontrar la representación más adecuada, facilitando el desarrollo de la aplicación. Variadas formas de representar o abstraer el contexto se han presentado. Dey *et al* (1999) muestra una arquitectura centrada en sensores que actualizan una lista de parámetros del sistema asociados a una probabilidad. Schilit *et al* (1993) hace uso de servidores de entorno, que manejan la información del contexto y la distribuyen a los múltiples sistemas cliente. El modelo de contexto presentado en Harter *et al* (1999) está basado en un modelo conceptual Entidad-Relación. El modelo es mantenido en una base de datos actualizable en tiempo de ejecución. Henricksen *et al* (2001) presenta un modelo de contexto orientado a objetos y basado en grafos, que permite modelar el contexto como un conjunto de entidades y relaciones.

4. Modelo de contexto para RA

Figura 4. Migración de Objetos en RA



[2] La palabra contexto se define como la información implícita en una situación cualquiera.

Para desarrollar el estudio del modelo de contexto propuesto, se presentará un sistema de Realidad Aumentada que ilustra cada una de sus características. Una vez presentado se definirá el modelo y se ejemplificará describiendo la información de contexto requerida por tal aplicación de RA. Para garantizar la generalidad del modelo de datos, éste se definirá formalmente.

4.1 Caso de estudio

Manejar información digital con nuestras propias manos agilizaría la manipulación de datos y facilitaría la compresión de la información. Una aplicación en particular consistiría en hacer que un objeto digital, tal como un archivo representado por un icono, se pudiera "extraer" del computador. Una vez en nuestras manos tendríamos la oportunidad de mover el objeto a cualquier lugar físico y observarlo en el espacio. Este tipo de interacción haría más fácil e intuitivo trasladar un archivo de una máquina a otra, o compartir información más fácilmente en un grupo de trabajo. La Figura 4 ilustra el caso propuesto. En a) un usuario tiene en su computador un grupo de archivos, representados en este caso por objetos tridimensionales. En b) gracias al sistema de RA que está al tanto del contexto, el usuario toma el objeto de su computador. El objeto adquiere una representación en el mundo real gracias a la RA. Tal objeto puede ser conservado en la mano o computador móvil del usuario. En c) el usuario enseña a sus compañeros el objeto. En el caso de una gráfica tridimensional sus compañeros podrían hacer comentarios acerca de su diseño. En caso de un archivo de sonido otros usuarios podrían escucharlo a medida que se acercan. En d) el usuario decide retornar el objeto en una estación de trabajo distinta. Tal sistema de RA debe estar "consciente" de la situación en que se desenvuelve el usuario. Este sistema debe conocer, por lo menos, la ubicación del usuario, dispositivos a los cuales tiene acceso y ubicación de los objetos virtuales.

4.2 Definición del modelo

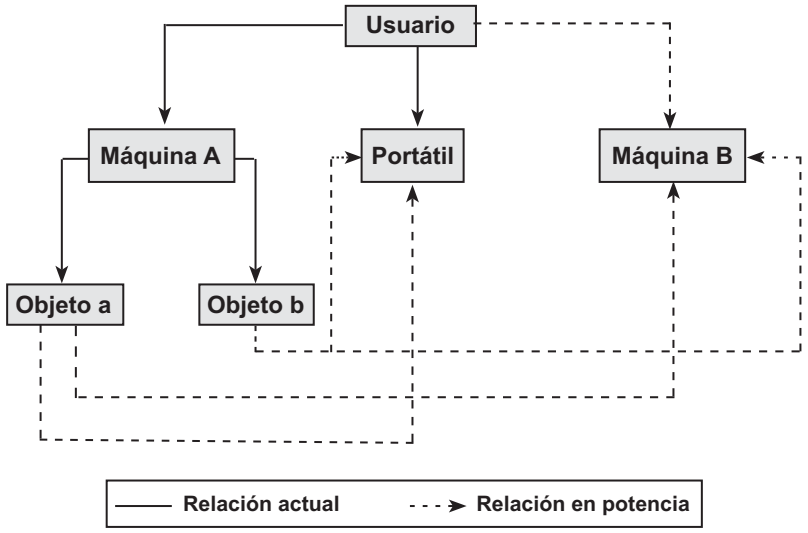
Se requiere un modelo de datos para expresar la situación en la que la aplicación de realidad aumentada toma lugar. Tal modelo debe incluir los elementos virtuales, los elementos del contexto y las relaciones entre éstos. Existen relaciones entre los elementos, tales como posesión, uso o conocimiento. Se presenta un modelo de datos orientado a objetos, en el cual la información del contexto requerida por la aplicación es estructurada en un conjunto de objetos organizados en forma de grafo, describiendo entidades reales o virtuales. Los objetos físicos pueden representar objetos o situaciones del contexto. Un objeto físico podría ser un lugar o una actividad (Ej: estudiar). Los objetos virtuales son elementos de información (procesos o datos). Las entidades se relacionan con otras entidades por medio de enlaces dirigidos. Las entidades y relaciones cuentan con atributos, estos especifican, por ejemplo el nombre, características o valores numéricos.

El modelo es simple y flexible. Esto permite su uso en múltiples aplicaciones de RA. Este modelo no restringe los tipos de relaciones entre las entidades, de esta forma se garantiza la potencia de representación y el soporte a distintos tipos de contexto. Como se definirá a continuación, cada aplicación tiene la posibilidad de restringir los tipos de entidades y relaciones a un conjunto limitado por medio de esquemas.

4.3 Componentes del modelo

Para mostrar los componentes del modelo se modelará la situación del caso de estudio. La Figura 5 presenta un primer acercamiento al modelo del sistema. Las entidades participantes son: el usuario, los dispositivos (tanto computadores de escritorio como el computador móvil del usuario) y los objetos virtuales. Existen relaciones establecidas que no varían en el tiempo, por ejemplo la relación del usuario con su equipo portátil permanecerá durante la ejecución de la aplicación. Existen relaciones que cambian durante la ejecución, tal como la relación usuario-máquina. En el gráfico se presentan relaciones en potencia, como las relaciones que podrían establecerse durante la ejecución. Los objetos, cuando se trasladan de máquina, adquieren una relación con la máquina en la que se encuentran y abandonan la relación anterior.

Figura 5. Primer acercamiento al modelo



4.3.1 Entidades

Una entidad es cualquier objeto físico o virtual. La definición de objeto representa aquí el mismo concepto de objeto de la programación Orientada a Objetos. Una entidad podría ser una persona, un lugar, una actividad o un sistema. De igual forma que un objeto, toda entidad pertenece a una clase de entidades. Para un usuario, la clase de su entidad es *Persona*. Toda entidad puede o no relacionarse con otras entidades por medio de relaciones. Además, toda entidad puede caracterizarse por atributos que son a su vez propios de cada entidad y únicos. Un atributo sería, para un dispositivo, por ejemplo, su tipo o especificaciones. Otros atributos podrían tomar distintos valores y tipos de datos básicos.

En el ejemplo, *máquina A*, *máquina B* y portátil son entidades de la clase de objetos *Dispositivo*. Los objetos a y b pertenecen a la clase *Objeto*[3]. Por medio de atributos se les dará características a los objetos a y b. En este caso a y b tendrían un atributo llamado *tipo*, que tomaría el valor de *virtual*.

[3] Objeto toma aquí la connotación de objeto tangible u objeto 3D y no en el sentido de la programación orientada a objetos.

4.3.2 Relaciones

Una relación es un enlace unidireccional entre entidades. Una relación es una afirmación acerca de la entidad en la que se origina; por ejemplo la relación entre *usuario* y *máquina A*, puede tomarse como la afirmación “usuario hace uso de máquina A”. Los enlaces son dirigidos para dar significado a tales relaciones. Las relaciones son el elemento fundamental del modelo, pues describen el contexto. Debe insistirse en que las relaciones no son únicamente espaciales. Puede presentarse por ejemplo una relación trabaja con entre dos usuarios. El valor semántico de estas relaciones es dado por el diseñador y la aplicación. Por tanto, existen múltiples tipos de relaciones, cada una útil de acuerdo con la aplicación.

En general las relaciones pueden ser estáticas o dinámicas durante el tiempo de ejecución. Las relaciones estáticas permanecen invariables. Las relaciones dinámicas, por su parte, cambian a medida que se desarrolla la situación. Las relaciones también cuentan con atributos que describen características sobre sí mismas. Estos atributos pueden usarse para contener las características temporales y de imperfección del contexto.

Una relación dinámica puede cambiar tanto en la entidad a la que apunta como en los valores de sus atributos. Las relaciones dinámicas son alteradas por decisión de la aplicación, valores derivados de un sensor o tiempo. En el modelo, la relación usuario portátil es una relación estática. La relación usuario máquina es una relación dinámica determinada por un sensor de ubicación. Un atributo propio de ésta sería la probabilidad o el error asociado a la posición del usuario. Las relaciones *objeto a* y *b* con *máquina B* y *portátil*, son relaciones modificadas a decisión del usuario por medio de interacción con el sistema.

4.4 Definición formal

El fundamento teórico para dar soporte a este modelo de datos se encontró en los sistemas de bases de datos orientados a objetos. Los sistemas de bases de datos; requieren de igual forma una definición formal que permita verificar su validez en cuanto a expresividad y computabilidad. Los sistemas de bases de datos orientados a objetos son descritos por Atkinson *et al* (1989) y Graham (1991).

El modelo de datos orientado a objetos de Gyssens *et al* (1994) tiene la potencia suficiente para cubrir el modelo propuesto. El modelo se especifica como un multigrafo dirigido, que puede ser restringido por un esquema y manipulado por medio de un conjunto básico de operaciones. En el grafo, tanto entidades como atributos son definidos como nodos. Para permitir la adición de atributos a las relaciones, éstas son descompuestas en un par de aristas y un nodo.

La definición se divide en esquema e instancia. Un esquema es una restricción sobre un modelo de datos, que permite tanto al diseñador como a la aplicación tener un conjunto de reglas válidas que describen una estructura bien formada en un lenguaje o representación gráfica. Una instancia es una construcción válida de acuerdo con un esquema.

Se definen a continuación esquemas, instancias y operaciones para el modelo.

Un esquema se define como una cinco-tupla $S = (C, A, F, NF, \mathcal{P})$ *donde:*

- C *es un conjunto finito de nombres de clase de objetos;*
- A *es un conjunto finito de nombres de tipo de atributo de objetos;*
- F *es un conjunto finito de nombres tipo de enlaces funcionales;*
- NF *es un conjunto finito de nombres de tipo de enlaces no-funcionales; y*
- $\mathcal{P} \subset C \times (F \cup NF) \times (C \cup A)$.

Un esquema puede ser representado por un grafo dirigido con dos tipos de nodos y dos tipos de aristas. El primer tipo de nodos toma cada uno de los nombres en C. *El segundo tipo de nodos toma los nombres en* A. *El primer tipo de aristas están marcadas con los nombres en* F *y el segundo tipo marcado con nombres en* NF. $\mathcal{P}$ *representa las aristas en el grafo.*

4.4.1 Definición del esquema

Las Figura 6 enseña el esquema para el caso. Las aristas funcionales representan propiedades y relaciones entre clases únicas. Las aristas no-funcionales, sin embargo, expresan multiplicidad de relaciones. Los nodos *usuario* y *dispositivo* son ejemplos de clases de entidades. Nótese cómo las relaciones *usa*, *tiene* y *porta* se representan también como nodos, permitiendo que tengan atributos. Este esquema define que la relación *tiene* es posible solamente entre un *usuario* y un *dispositivo*. La arista entre *tiene* y *dispositivo* es funcional, para evitar enlaces con otro *dispositivo.* Las relaciones entre entidades en este esquema (*usuario*, *dispositivo* y *objeto*), sólo se dan por intermedio de nodos de relación (*usa*, *tiene* y *porta*).

Las aristas de los atributos tienen valores semánticos. Los atributos son tipos básicos de lenguaje (se puede considerar *archivo* como un tipo básico). El uso de aristas funcionales para los atributos asegura que no habrá atributos duplicados.

Figura 6. Esquema para el caso de estudio

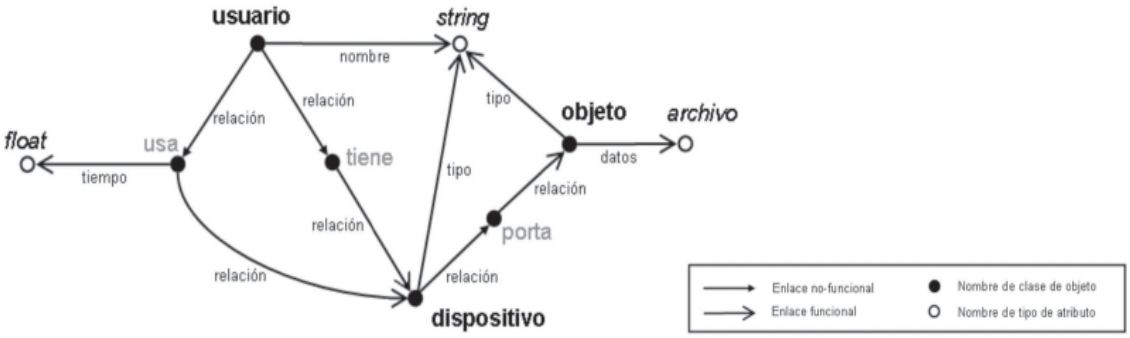
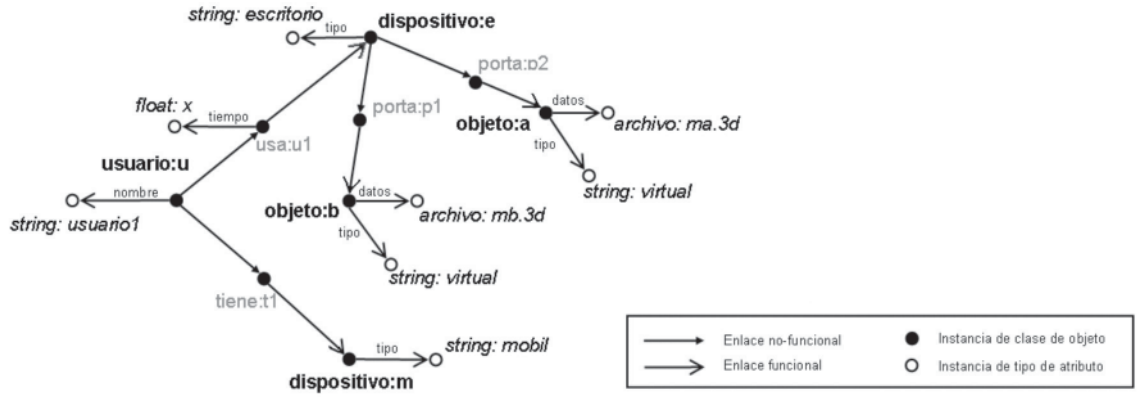


Figura 7. Instancia para el caso de estudio



4.4.2 Definición de las instancias

Sea $\mathcal{S} = (C, A, F, NF, \mathcal{P})$ un esquema. Una instancia sobre el esquema es un grafo $\mathcal{I} = (N, E)$ donde:

- *N es un conjunto finito de nodos con nombre. Para un nodo n en N, la función de obtención de tipo $\lambda(n)$ debe estar en $C \cup A$; si $\lambda(n)$ se encuentra en C se dice que n es un objeto; si $\lambda(n)$ se encuentra en A, se habla de n como un atributo;*
- *E es un conjunto de aristas con nombre. Para una arista e en E, se cumple que $e = (m, \alpha, n)$ con m y n en N y $\alpha = \lambda(e)$ en $F \cup NF$; además $(\lambda(m), \alpha, \lambda(n)) \in \mathcal{P}$; si $\lambda(e)$ está en F, e se define como una arista funcional; si $\lambda(e)$ está en NF se refiere a una arista no-funcional; y*
- *Si (m, α, n_1) y $(m, \alpha, n_2) \in E$, entonces $\lambda(n_1) = \lambda(n_2)$, (las clases de objetos conectados por aristas de tipo α a un mismo nodo m deben ser iguales para conservar el esquema); además, si $\alpha \in F$, entonces $n_1 = n_2$ (si α es funcional no se permite que sea duplicada).*

La Figura 7 ejemplifica una instancia sobre el esquema propuesto en la Figura 6. Las clases y tipos de atributos toman valores concretos de objetos y datos al convertirse en instancias[4]. Sólo las relaciones definidas por el esquema pueden establecerse. Un usuario puede contar con múltiples dispositivos y éste a su vez con múltiples objetos, dado que el enlace es no-funcional.

4.4.3 Operaciones

Se definen cinco operaciones sobre el modelo:

- Adicionar un nodo
- Adicionar una arista

[4] Compárese con la relación clase-objeto en programación orientada a objetos.

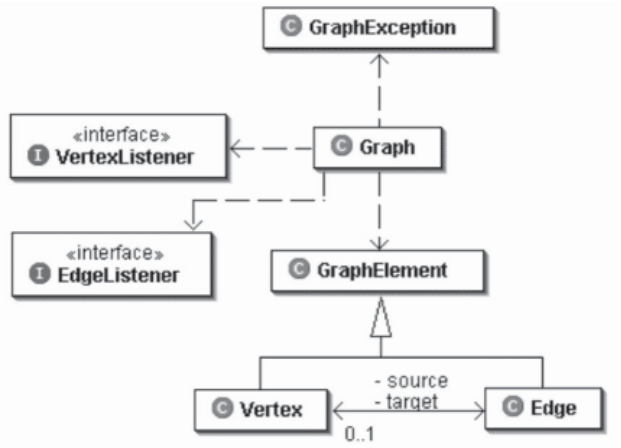
- Remover un nodo
- Remover una arista
- Redirigir una arista

Las primeras cuatro operaciones son básicas en teoría de modelos de grafos. La operación de redirección de arista consiste en eliminar y adicionar una arista. Las operaciones son válidas dentro del modelo. Esto puede verificarse en Gyssens *et al* (1994).

5. Implementación

Se implementó el sistema propuesto en el caso de estudio. La implementación permite evaluar las ventajas del uso de este modelo frente a los métodos de implementación tradicionales como se verá en la sección 6. Ésta es útil para probar el concepto propuesto. A continuación se presentará la implementación del modelo y la arquitectura del sistema, y cómo encaja el modelo dentro de ésta.

Figura 8. Librería de grafos



5.1 Representación del modelo

Se implementó una librería de grafos dirigidos en el lenguaje Java para representar el modelo. La librería contiene la funcionalidad básica de construcción de entidades, relaciones y atributos. El diagrama de objetos de la Figura 8 enseña las clases principales de la librería. La clase *Graph* es la clase principal. Una instancia de *Graph* contiene un grupo de vértices (nodos[5]) y aristas formando un grafo dirigido. En la Figura 8 las líneas punteadas señalan dependencias de *Graph* con las demás clases. *VertexListener* y *EdgeListener* son interfaces que deben implementarse para ser subscriptores de un objeto que se encarga de notificar ante cambios en los vértices o aristas. Los vértices y aristas son representados por los objetos *Vertex* y *Edge*.

Graph implementa las cinco operaciones principales definidas en 4.4. Además adiciona funciones para consultar el estado del grafo, obtener elementos y adicionar subscriptores. Los elementos del grafo tienen en común atributos, identificación y datos de usuario. Los atributos se definen como un par nombre-valor almacenados ambos como objetos *String* de Java.

La clase *Vertex* representa un vértice o nodo. La clase *Edge* representa una relación. Un objeto *Edge* representa un nodo de relación más no un enlace. Los enlaces no requieren de representación debido a la descomposición de la que se habló en 4.4. Los valores semánticos de la relación son almacenados por un atributo especial en *Edge*.

En la implementación para la aplicación del caso de estudio, se requirió que el grafo se representara en XML para su almacenamiento. Puede pasarse del grafo XML al grafo en objetos y viceversa. Se usaron elementos *node* para representar vértices y *edge* para representar relaciones. Un atributo de identificación *id* es necesario para cada vértice y relación. La clase o tipo de relación se almacena como *class*. Los enlaces se representan como los atributos *source* y *target* que apuntarán a los identificadores de los nodos. Los demás atributos son de igual forma atributos XML.

5.2 Arquitectura del sistema

La aplicación propuesta en el caso de estudio se construyó como un sistema distribuido

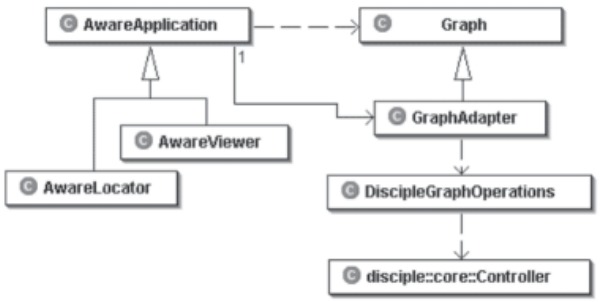
[5] Se nombraron vértices en vez de nodos par evitar conflictos de nombre con otras librerías de Java.

compuesto, por tres sub-aplicaciones conectadas por un *middleware* común. Para su construcción se partió de *Parallel Worlds Framework*, re-implementando el módulo de Realidad Aumentada, agregando soporte a una interfaz de usuario para escritorio y nuevos métodos de ubicación. El grafo en forma de *Scene Graph* original se usó a manera de jerarquía plana, donde los vértices y aristas son todos hijos de la raíz del árbol.

Como se presentó en 4.1, la aplicación permite a un usuario llevar un objeto virtual de un equipo de escritorio a la realidad y luego tomarlo de regreso al mismo equipo de escritorio u otro. La ubicación del objeto, es decir, aquello que lo contiene, es el escritorio o el computador móvil. Se requiere entonces:

- Presentar al usuario el objeto virtual en el computador de escritorio.
- Presentar al usuario el objeto virtual en la realidad.
- El sistema debe saber en qué equipo se encuentra el usuario.

Figura 9: Arquitectura general del sistema



En la Figura 9 puede verse la arquitectura general del sistema. Se definió la clase *AwareApplication* para encerrar la funcionalidad común de las denominadas "sub-aplicaciones". La función más importante de *AwareApplication* es el acceso al modelo, eso se hace con un objeto *GraphAdapter* que hereda de *Graph*.

GraphAdapter oculta los detalles de implementación inherentes al *middleware* de PWF, DISCIPLE. *DiscipleGraphOperations* tiene la tarea de convertir las operaciones sobre el grafo en operaciones de manipulación del SG de DISCIPLE y viceversa. Una vez convertidas en operaciones del SG, el *middleware* se encarga de la distribución y sincronización de las operaciones en la red. Las operaciones en el sentido opuesto son notificadas al *GraphAdapter*, que usa los métodos de publicador-subscriptor del objeto grafo.

Pueden extenderse dos tipos de sub-aplicaciones, una aplicación de presentación y una de localización. Una aplicación de presentación (*AwareViewer*) genera gráficas de realidad aumentada. La aplicación de localización (*AwareLocator*) controla el sistema que ubicación del usuario. Se describe ahora cada una en detalle

5.2.1 Visores

Se implementó el visor heredando de *Aware-Application*. La presentación se hace independiente con la interfaz *Viewer*, que se implementa de acuerdo a si es un visor de RA (*AugmentedViewer*) o de escritorio (*DesktopViewer*). *AugmentedViewer* hace uso de un adaptador que esconde el funcionamiento del representador gráfico de RA. Este se encuentra implementado en código nativo para mayor rendimiento[6]. *DesktopViewer* extiende una ventana gráfica de las librerías de Java.

El uso del modelo hizo de la construcción del visor algo bastante simple. Los visores de escritorio presentan un botón que permite al usuario tomar o devolver el objeto. Descargar o tomar el objeto es una operación de redirección de enlace entre un objeto virtual y el dispositivo que lo porta. Obtener conocimiento de qué dispositivos tiene el usuario se realiza con una simple consulta al nodo que lo representa. Una vez ubicado el dispositivo del usuario, los objetos se trasladan allí. Tanto el visor de realidad aumentada como

[6] Más detalles sobre la implementación pueden verse en Ganapathy et al (2003) y Kato y Billinghurst (1999).

el de escritorio son notificados de los cambios y determinan si tienen o no el objeto virtual.

El activar o desactivar el botón se determina cuando el localizador decide en qué lugar se encuentra el usuario. Si un visualizador de escritorio es notificado de un cambio en la relación que indica la ubicación del usuario, éste verifica si se trata del computador donde se está ejecutando, en caso de que sí, debe activarlo, pues el usuario acaba de llegar. De lo contrario debe desactivarlo, pues no está allí. El visor de realidad aumentada, que se ejecuta en el computador móvil del usuario, tiene la función de estar atento a cambios en la ubicación de los objetos virtuales y presentar los objetos 3D. Si, según el modelo, la ubicación de los objetos virtuales es el computador móvil, estos deben presentarse en RA.

5.2.2 Localizador

El localizador se encarga de determinar qué equipo se encuentra utilizando el usuario o si el usuario no se encuentra en ninguno. Se usa el sistema de ubicación de PWF. La ubicación se hace con una cámara de video y patrones gráficos que pueden ser reconocidos por ARToolkit. La lógica del localizador se hace bastante simple con el uso del modelo. Al sistema de ubicación se le asigna un conjunto de patrones y a cada patrón se le asigna una ubicación. Una vez se reconoce un patrón se obtiene el nombre de la ubicación. El sistema de ubicación notifica al *AwareLocator* cuya reacción es ejecutar una operación de redirección de arista entre la entidad que representa el usuario y la entidad del equipo de escritorio que representa la ubicación. El localizador asocia un atributo de tiempo a la relación entre el usuario y el equipo. Este atributo es útil para comprobar cuándo estuvo el usuario en un lugar, para poder mantener temporalmente una relación en caso de que falle el sistema de ubicación.

6. Evaluación

Las ventajas encontradas en el uso de un modelo de datos que contenga información del contexto, pueden ser evaluadas en comparación con el modelo de los sistemas tradicionales. Se hizo un análisis preliminar de las desventajas teóricas del modelo previo. Se realizó luego una evaluación práctica. La evaluación se hizo mediante una prueba de diseño y observación en los beneficios de desarrollo con el nuevo modelo. Finalmente se comprobó el cumplimiento de los requerimientos establecidos en la sección 3.3.

6.1 Comparación con el modelo tradicional

La preferencia por el *Scene Graph* como modelo central proviene de las aplicaciones de gráficas tridimensionales y la Realidad Virtual (ver secciones 2.1 y 2.2). El modelo del SG es limitado y debe complementarse con otros modelos que expresen en mejor medida la situación del contexto. Como se mostró en 4.2, el contexto puede representarse como un conjunto de relaciones entre entidades u objetos que corresponden a elementos de la realidad. Los modelos de entidades y objetos son usados en sistemas que manejan datos complejos, tales como la Inteligencia Artificial (Graham, 1991). Para Henricksen *et al* (2002) y como se definió en la sección 4.4, las relaciones entre entidades del mundo son relaciones semánticas de múltiples tipos y en sí expresan la información del contexto. En el modelo de SG en su forma original (Strauss y Carey, 1992), las relaciones son de tipo jerárquicas geométricas. El *Scene Graph* no puede, por sí solo, representar la información del contexto.

6.2 Experimento de diseño

Se realizó un experimento de diseño para comparar los modelos de datos. El experimento consistió en diseñar el mismo sistema propuesto en el caso de estudio con modelos de datos restringidos. Se hizo un estudio por niveles, en el que en cada nivel se agregaba más información de contexto.

La primera aproximación, o nivel cero, fue el caso de una aplicación de escritorio con la misma funcionalidad, es decir, que permitiera trasladar

un objeto. La conclusión para este diseño fue que sería una aplicación habitual de transferencia de archivos, donde la información de la situación es provista por el usuario manualmente. Esta información, como cuál es el destino del archivo, es dada explícitamente por el usuario. La aplicación sólo cuenta con el modelo de datos del objeto, que corresponde al sistema de archivos y desconoce totalmente el entorno y el usuario. Este sistema es similar a una aplicación actual como FTP.

Para el nivel uno se consideró un sistema de realidad aumentada que mantuviera la información de los objetos, pero sin agregar aún la información del usuario. La implementación se basó en el modelo de datos de PWF (véase sección 2.1), pues al igual que en este nivel, PWF sólo consideraba en el modelo los objetos reales, virtuales y sus relaciones geométricas.

En la aplicación de nivel uno, debido a que no se contaba con la información del usuario, se desconocía su ubicación y no se podían determinar los estados del visor. Para fijar la posición del objeto en este nivel, cada aplicación asociaba la posición virtual de los objetos virtuales con el lugar en que se encontraban. Es decir, que para estar en un computador de escritorio el objeto debía estar dentro de un rango de coordenadas (un cubo), fuera de este cubo el objeto estaba en RA. Este tipo de asociación se ha usado previamente en Realidad Aumentada en el trabajo de Schmalstieg *et al* (2002).

En el nivel dos se consideró la ubicación del usuario, aunque no se agregó el usuario como un elemento del modelo. Para determinar la ubicación de igual forma se usaron relaciones espaciales entre la posición del usuario y la de los equipos. Trató de darse una posición aproximada del usuario usando el reconocedor de patrones del localizador. Esta aplicación es semejante a la de la arquitectura de referencia presentada en la sección 2.2. Al igual que en ésa, una parte del sistema (subsistema de contexto) se encarga de recoger la información de los sensores de ubicación para manejar la información del contexto. Contando con la posición se podía entonces dar conocimiento a la aplicación de donde se encontraba el usuario y dar el mismo funcionamiento al visor. A pesar de que el sistema cumplía con la misma función, resultó menos intuitivo al momento de implementarlo. El mapeo de la posición del usuario no tenía sentido sabiendo que lo único que se requería era saber en cual equipo se encontraba, y no su posición exacta. Otra desventaja del modelo en este punto, es la ausencia del atributo de tiempo que puede tomar la relación *usa.* Tal valor se usa para saber qué tan válida puede ser la afirmación “usa”, en caso de que fallara el sistema de ubicación. Las relaciones en un modelo SG no incluyen atributos.

Se determinó que el nivel tres, que agregaría al usuario como un elemento, correspondería a un modelo similar al propuesto sin los valores semánticos ni atributos de las relaciones. Este modelo no se evaluó por sus similitudes y porque se salía de la definición de SG, resultando en un modelo híbrido. El modelo propuesto en este trabajo sería de nivel cuatro, que agrega atributos a las aplicaciones y por tanto valores semánticos.

Se concluyó que la aplicación como tal resultaba más compleja de desarrollar con el modelo tradicional. Este modelo no integra en una misma estructura la información del contexto (incluyendo al usuario) y limita las relaciones a relaciones espaciales. Se concluyó, además, que en caso de implementarse puede conservarse el modelo gráfico debido a su eficiencia, pero debe existir un modelo de datos de contexto como se presentó aquí.

6.3 Ventajas de codificación

Se analizaron las ventajas del modelo al momento de la programación. Haciendo uso del modelo, la lógica del sistema se redujo a comprobar los estados del modelo y reaccionar por medio del conjunto de operaciones básicas. Las cinco operaciones básicas fueron las únicas requeridas para cambiar el estado del modelo. Los objetos de un SG en general no cuentan con identificadores, por lo que la

operación de adición requiere de una búsqueda. En la implementación del modelo para el caso, la misma operación se hace por medio del objeto grafo. Todos los elementos del grafo cuentan con un identificador, y se usa éste para adquirir el objeto y cargar el objeto 3D. El grafo mantiene una tabla de *hashing* con cada uno de los elementos y relaciones. El manejo de identidades fue una de las mayores ventajas encontradas.

Se encontraron ventajas también con la redirección de aristas. La función del localizador es informar al sistema dónde se encuentra el usuario, en este caso en qué equipo. Una vez el localizador conoce el nombre de la posición, adquiere la arista que relaciona al usuario con la ubicación y el vértice correspondiente. Una operación de redirección es suficiente para determinar la ubicación del usuario. En el *Scene Graph*, debido a que se hace uso de geometría, la misma operación requeriría determinar la posición en el espacio del usuario y, basándose en ésta, calcular en qué lugar se ubica.

6.4 Satisfacción de los requerimientos

Se comprobará la satisfacción de los requerimientos propuestos en la sección 2.3. El requerimiento (1) se cumple con la definición de clases y objetos en 4.4. Un objeto puede abstraer cualquier tipo de entidad, virtual o real, y almacenar sus atributos. La unicidad de (2) se cumple con la definición del modelo en forma de grafo y las operaciones sobre éste. El grafo es un ente único. Sólo hay un objeto grafo por aplicación. La escalabilidad y extensibilidad requeridas en (3) son permitidas por la multiplicidad de relaciones definida en 4.4. La capacidad de abstracción por el sistema (4) es evidente pues, la estructura de grafo es representable y se pudo implementar en un lenguaje de programación como Java, como se vio en la sección 5. Los grafos son una estructura natural para los humanos (Grassman y Tremblay, 1996) y han sido una abstracción útil para variedad de problemas (Aho y Ullman, 1995). La definición formal del modelo exigido por (5) se presentó en 4.4 y está soportada por Gyssens et al (1994).

7. Aporte de diseño

El aporte al diseño de aplicaciones RA se presenta como un Patrón de Diseño. Un patrón de diseño describe una solución a un problema de software, que puede reutilizarse. El patrón resumirá la propuesta en una forma que sea de rápida comprensión y reutilización. Se hace uso de la estructura de definición de patrones presentada por Gamma *et al* (1995).

NOMBRE:
Modelo de Contexto.

CLASIFICACIÓN:
Estructural - Objetos.

OBJETIVO:
Representar la información de la situación del mundo dentro de la aplicación de Realidad Aumentada.

MOTIVACIÓN:
Las aplicaciones de Realidad Aumentada requieren conocer la situación en que se encuentra el usuario. Tal situación puede incluir el usuario, objetos virtuales y reales, dispositivos y demás elementos relevantes a la aplicación. El modelo gráfico de la Realidad Virtual es en general insuficiente para almacenar tal información.

APLICABILIDAD:
Haga uso del patrón de Modelo de Contexto cuando existan relaciones más que geométricas entre los elementos del mundo real y virtual.

ESTRUCTURA:
Separe la aplicación del modelo. Maneje el modelo como un objeto que a su vez encierra un conjunto de elementos. Los elementos representan las partes del mundo. Estos están relacionados entre sí.

PARTICIPANTES:
Application: Contiene la lógica de la aplicación.
Model: Abstrae el modelo.
ModelElement: Abstrae los elementos del modelo.

COLABORACIÓN:
Application actúa sobre *Model*. *Model* efectúa los cambios sobre los *ModelElement*. Ante cambios en *ModelElement*, *Model* notifica a *Application*.

CONSECUENCIAS:
- Flexibilidad y riqueza de representación.
- Separación de la lógica, el modelo y la vista.
- Múltiples aplicaciones pueden acceder al modelo.
- Debe mantenerse sincronización entre la vista y el modelo.

IMPLEMENTACIÓN:
- Se debe proveer una vista para el modelo. Cada aplicación debe encargarse de esto.
- El modelo debe proteger los elementos de modificación externa de las relaciones entre elementos.
- Defina el conjunto de operaciones básicas sobre el modelo.

PATRONES RELACIONADOS:
Modelo-Vista-Controlador, Observador, Fachada.

Conclusiones

Este trabajo presentó un modelo de datos para representar la información del mundo o contexto, en aplicaciones de Realidad Aumentada (RA). Se definió la RA y se presentó la necesidad de un nuevo modelo de datos para representar la información del mundo. Para proceder con la propuesta de modelo se presentaron los problemas de las arquitecturas actuales, partiendo de la experiencia en el desarrollo de dos sistemas de Realidad Aumentada y presentando una arquitectura de referencia. Se definió una lista de requerimientos que debía cumplir el modelo de contexto. Luego se planteó el modelo y se expresó formalmente. El caso de estudio se implementó con el modelo de contexto y se presentó y evaluó el sistema construido.

Puede concluirse que la combinación real-virtual en la Realidad Aumentada es compleja y no consiste simplemente en sobreponer geométricamente un mundo sobre el otro. Las estructuras y modelos de datos no han sido de especial preocupación para el área de RA. Las arquitecturas de Realidad Aumentada actuales no cuentan con la información adecuada del mundo real. Debe considerarse que la información de contexto exhibe distintas características temporales, de imperfección, representación e interrelación. Un modelo de datos para RA debe admitir la representación de entidades virtuales, reales y sus relaciones; debe ser unificado; debe poder escalarse y extenderse; el modelo debe ser comprensible por el sistema y el diseñador; el modelo debe también poder expresarse formalmente.

El contexto puede representarse como un grafo formado por un conjunto de entidades y sus relaciones. La implementación del caso de estudio demostró la aplicabilidad del modelo y presentó sus características fundamentales. Se encontró que el *Scene Graph* no puede, por sí sólo, representar la información del contexto. El modelo no niega el uso del *Scene Graph*, sólo se sugiere el uso del modelo como modelo principal de la aplicación de RA.

A futuro se pretende realizar un estudio más profundo de modelos de datos, y en especial explorar la combinación de modelos de datos semánticos y orientados a objetos. También se busca encontrar una forma de comparación formal que permita expresar más estrictamente las ventajas del modelo de contexto frente al *Scene Graph*. Debe probarse el modelo con aplicaciones más complejas, de mayor interacción con el usuario y dinamismo para determinar su verdadero potencial. Existe un área en específico que es de gran atención. La sincronización entre el modelo de datos de contexto y el modelo de representación visual debe ser tenida en cuenta. Una instancia de modelo incluye tanto elementos visibles como no visibles. Los elementos visibles a ser representados en realidad aumentada podrían necesitar ser extraídos del modelo y organizados de acuerdo con las relaciones espaciales existentes.

El mapeo del modelo al grafo debe considerarse. En especial si los sistemas son complejos, altamente dinámicos y requieren gran cantidad de elementos gráficos. Se trata de un problema de bases de datos dual y fue uno de los problemas de diseño que intentó solucionar el SG en sus orígenes. El SG, por esta razón, mezcló el modelo y la vista para simplificar y tratar de hacer más eficiente el despliegue gráfico. El patrón Modelo-Vista-Controlador sugiere, sin embargo, tal separación. Ésta se hace bastante apropiada para sistemas distribuidos y que presentan la información de distintas formas, tales como los dependientes del contexto (Banavar *et al,* 2000). Para tal caso, formas de mapeo de modelo gráfico al modelo de datos deben investigarse. Métodos heurísticos o de extracción de árboles, como los algoritmos de árboles de expansión (Grassman y Tremblay, 1996), pueden tenerse en cuenta.

Bibliografía

Aho, Alfred y Ullman, Jeffrey (1995). "Foundations of Computer Science". En: *Computer Science Press.*

Atkinson, Malcolm, Bancilhon, François, DeWitt, David, Dittrich, Klaus, Maier, David, y Zdonik, Stanley (1989). "The object-oriented database system manifesto". En: *Proceedings of the First International Conference on Deductive and Object-Oriented Databases*, Kyoto, Japón. pp. 223–240.

Azuma, R. (1995). "A survey of augmented reality". En: *Computer Graphics (SIGGRAPH '95 Proceedings, Course Notes 9: Developing Advanced Virtual Reality Applications)*, pp. 1–38.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., y MacIntyre, Blair (2001). "Recent advances in augmented reality". En: *IEEE Computer Graphics and Applications*.

Banavar, Guruduth, Beck, J., Gluzberg, E., Munson, J., Sussman, J. y Zukowski, D. (2000), "Challenges: an application model for pervasive computing". En: *Proceedings of the 6th annual international conference on Mobile computing and networking*, ACM Press. pp. 266–274.

Brügge, Bernd, MacWilliams, Asa y Reicher, Thomas (2002). "Software architectures for augmented reality systems – report to the ARVIKA consortium". En: *Technical Report.* Technische Universität München. TUM-I0410.

Dey, A. K. y Abowd, G. D. (2000). "Towards a better understanding of context and context-awareness". En: *CHIA'00 workshop on Context-Awareness*.

Dey, A., Abowd, G., y Salber, D. (1999). "A context-based infrastructure for smart environments". En: *Proceedings of the 1st International Workshop on Managing*

Interactions in Smart Environments (MANSE '99). pp. 114–128.

Gamma, E., Helm, R., Johnson, R., y Vlissides, J. (1995). "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley.

Ganapathy, S. Kicha, Morde, Ashutosh, y Agudelo, Andres (2003). "Tele-collaboration in parallel worlds". En: *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence.* ACM Press. pp. 67–69.

Graham, Ian. (1991). *Object oriented methods*. Addison-Wesley Longman Publishing Co.

Grassman, Winfried K. y Tremblay, Jean-Paul (1996). *Logic and Discrete Mathematics*. Prentice-Hall.

Gyssens, Marc, Paredaens, Jan, den Bussche, Jan Van, y van Gucht, Dirk (1994). "A graph-oriented object database model". En: *IEEE Transactions on Knowledge and Data Engineering.*

Harter, A., Hopper, A., Steggles, P., Ward, A., y Webster, P. (1999). "The anatomy of a context-aware application". En: *Mobile computing and networking*, pp. 59 – 68.

Henricksen, K., Indulska, J. y Rakotonirainy, A. (2002). "Modeling context information in pervasive computing systems". En: *1st International Conference on Pervasive Computing*, Springer. Suiza.

Henricksen, Karen, Indulska, Jadwiga y Rakotonirainy, Andry (2001). "Infrastructure for pervasive computing: Challenges". En: *GI Jahrestagung (1)*, pp. 214–222.

Hull, R., Neaves, P., y Bedford-Roberts, J. (1997). "Towards situated computing". En: *1st International Symposium on Wearable Computers*, pp. 146–153.

Kato, H. y Billinghurst, M (1999). "Marker tracking and hmd calibration for a video-based augmented reality conferencing system". En: *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality '99*, pp. 85–94.

Kindberg, T. *et al*. (2000). "People, places, things: Web presence for the real world". En: *Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'00)*, pp. 19-??.

Marsic, Ivan (1999). "DISCIPLE: a framework for multimodal collaboration in heterogeneous environments". En: *ACM Comput. Surv.*, Vol. 31. No. 2. p. 4.

Newman, Joseph, Wagner, Martin, Bauer, Martin, MacWilliams, Asa, Pintaric, Thomas, Beyer, Dagmar, Pustka, Daniel, Strasser, Franz, Schmalstieg, Dieter, y Klinker, Gudrun (2004). "Ubiquitous tracking for augmented reality". En: *Proceedings of The Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, Valtimore, EU.

Norman, Donald. (1998). *The Invisible Computer*. Massachusetts: MIT Press

Reicher, Thomas, MacWilliams, Asa, Brügge, Bernd, y Klinker, Gudrun (2003). "Results of a study on software architectures for augmented reality systems". En: *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality*.

Reitmayr, Gerhard (2004). "On Software Design for Augmented Reality". Tesis. Vienna University of Technology.

Satyanarayanan, M. (2001). "Pervasive computing: Vision and challenges". En: *IEEE Personal Communications*, pp. 10–17.

Schilit, B., Adams, N. y Want, R. (1994). "Context-aware computing applications".

En: *1st International Workshop on Mobile Computing Systems and Applications*, pp. 85-90.

Schilit, Bill, Theimer, Marvin, y Welch, Brent (1993). "Customizing mobile application". En: *USENIX Symposium on Mobile and Location-independent Computing*, pp. 129–138. Cambridge, MA, EU.

Schmalstieg, Dieter, *et al*. (2002). "The studierstube augmented reality project". En: *Presence: Teleoper. Virtual Environ.* Vol. 11. No. 1. pp. 33-54.

Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., y Van de Velde, W. (1999). "Advanced interaction in context". En: *Lecture Notes in Computer Science*, 1707. pp. 89-??.

Strauss, Paul S. y Carey, Rikk (1992). "An object-oriented 3d graphics toolkit". En: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ACM Press. pp. 341-349.

Weiser, Mark (1995). "The computer for the 21st century". En: *Human-computer interaction: toward the year 2000*, pp. 933-940.