Una arquitectura para sistemas de ubicación

Esteban Robles Luna

Trabajo final para obtener el grado de
Licenciado en Informática
de la
Facultad de Informática,
Universidad Nacional de La Plata,
Argentina

Director: Dr. Gustavo Rossi
Co-director: Dra. Silvia Gordillo
La Plata, Agosto de 2007

Índice general

Índice de figuras

Agradecimientos

El presente trabajo marca un punto importante en mi vida. Desde que ingrese a la facultad corrí tras un objetivo: recibirme de licenciado. Durante estos años y desde mi adolescencia he contado con varias personas que me han ayudado a formarme como persona y profesional.

En primer lugar quiero agradecer a mi familia: mis viejos y mi hermano han sido un apoyo incondicional desde el punto de vista emocional y económico. Sin su guía y afecto hubiera sido imposible realizar esta carrera. Gracias Gabi por acompañarme incluso cuando nuestras charlas se tornaban monotemáticas y pesadas.

Sole, vos me acompañaste desde los primeros días que me conociste y me das todo tu apoyo y amor para que pueda continuar cuando bajo los brazos. Gracias por tus sugerencias sobre este trabajo.

Mi familia 'más lejana' también fue vital para mi formación en una ciudad lejana. Fueron invaluables los consejos y ayudas de mi tío Carlos y familia. Su apoyo y contención durante los primeros años de la carrera es algo que jamás podré agradecerles. A Rodrigo, Florencia y Matilde por los momentos que hemos vivido juntos durante estos últimos años.

A Gustavo por haberme incluido dentro del grupo de Context Aware (CAG) del LIFIA en donde desarrolle el presente trabajo. A él y a Silvia por haber aceptado ser mis directores del presente trabajo. A los integrantes del CAG, en particular a Andrés, que siempre fue mi guía y a Cecilia por sus sugerencias del presente trabajo.

Siempre pienso que una persona llega a ser lo que es por las cosas que le ha tocado vivir. En mi caso particular hay un conjunto de personas que quiero agradecer ya que ellos sentaron las bases de lo que soy hoy. Me ayudaron a crecer desde un punto de vista humano y profesional brindándome toda su amistad y apoyo. A Rosana Rodríguez por su amistad, sus enseñanzas en matemáticas y su

incentivo para continuar esforzándome en la Olimpiada de Matemática. A Silvia García por haberme ayudado a levantar en un momento de mi vida cuando estuve a punto de abandonar todo. A Julio Paladino que me ayudo a darme cuenta que el esfuerzo vale y que todo es posible si se dedica el tiempo necesario para lograrlo. ¡Como se extrañan las charlas con vos Julito! Por último a Liliana Boemo de Belfiori que siempre consiguió los recursos para que pudiera viajar a todas las competencias de matemáticas, computación o atletismo durante mi secundario en el Instituto América.

Por último a los chicos: Chiara, Jero, Juan, Alex y Marian que aunque estemos distanciados hemos compartido muchos momentos y hemos desarrollado numerosas materias juntos. Las reuniones y momentos vividos serán recordados siempre.

Capítulo 1

Introducción

1.1. Motivación

En los últimos años se ha experimentado un cambio notorio en el consumo de productos tecnológicos por parte de la sociedad. En el caso particular de nuestro país, este cambio se ha hecho notar con el consumo masivo de teléfonos celulares por parte de niños, adolescentes y adultos. Este fenómeno no es un hecho aislado sino que a nivel mundial la sociedad se ha abocado a la compra de reproductores de MP3, notebooks, PDAs y Smartphones, entre otros. Desde el punto de vista tecnológico, estos dispositivos no dejan de ser computadoras de uso general o específico, algunas de las cuales poseen un poder de cómputo semejante a las mejores PCs de hace 3 años. Sus prestaciones puede ser aún mejoradas si podemos incorporarles periféricos de entrada salida. Esto permitiría sensar ciertos aspectos del medio como pueden ser la ubicación (utilizando un GPS), las condiciones climáticas (utilizando un web service y la ubicación del dispositivo) y la hora actual (sincronizando el reloj del dispositivo con el reloj de la región donde el usuario se encuentre ubicado) permitiendo conocer el ambiente físico del dispositivo.

Desde el punto de vista del software, estos dispositivos poseen las mismas aplicaciones que antes encontrábamos en una PC de escritorio. Un ejemplo de ello es una agenda que notifica al usuario sus citas. Pensemos por un instante como es su funcionalidad, las notificaciones de las citas podrían mostrarse fácilmente por pantalla mediante un diálogo que capture la atención del usuario. Si trasladamos esta misma aplicación a un dispositivo móvil, como por ejemplo un Smartphone, la funcionalidad descripta no alcanzaría para cumplir el objetivo. Esto ocurre debido a que las notificaciones en pantalla serían totalmente inútiles ya que no podríamos asegurar que el usuario se encuentre observandolá en el instante que esta ocurra. Como alternativa, las notificaciones podrían realizar-

se emitiendo un sonido cada vez que una cita ocurra. Esta solución tiene un problema conocido que es el hecho de disturbar al resto de las personas cuando nos encontramos en una reunión o un cine. Por este motivo deseamos que la aplicación sea un poco más 'inteligente' y que cuando nos encontremos en una situación que amerite atención o silencio, en vez de utilizar sonidos, utilice vibraciones.

Para que las aplicaciones puedan adaptarse a su ambiente es necesario que puedan sensarlo. El sentir el ambiente ayuda a resolver los problemas que antes no aparecían en las aplicaciones de escritorio y que surgen del hecho de trasladar la misma aplicación a un dispositivo móvil o de enriquecer una aplicación existente. Estos problemas han provocado el surgimiento de nuevas áreas dentro de las ciencias de la computación como es el caso de las aplicaciones Context Aware [8, 10]. Este tipo de aplicaciones adaptan su comportamiento en base a lo que son capaces de percibir del contexto. Los aspectos del contexto que estas utilizan son variados y pueden ir desde la ubicación de los usuarios hasta la hora o condiciones climáticas del lugar donde éste se encuentre ubicado. Por ejemplo, podríamos disponer de un teléfono Context Aware que cuando detecte que hemos ingresado a un cine cambie, en forma automática, el estilo de timbre. El mismo cambio podría ser aplicado cuando hemos iniciado una reunión con un cliente. Para ello la aplicación necesitaría más información del contexto que simplemente la ubicación del usuario. Podríamos pensar que la hora de la reunión, el lugar de la misma y si la persona con la cual se va a mantener la reunión se encuentra en la misma sala, serían factores necesarios para determinar si ésta ha iniciado.

Otro conjunto de aplicaciones han surgido del hecho de querer proveer navegación hipermedial en base a la ubicación física del usuario. Supongamos que la Catedral de La Plata posee una modelo de hipermedia que nos permite recorrer las diferentes zonas como el altar, la entrada y el campanario. También podemos ver la biografía de las personas que trabajaron en la construcción de cada una de estas zonas. Aprovechando este modelo de hipermedia, queremos que los usuarios que visiten físicamente la catedral puedan navegar por dicho modelo desde sus dispositivos móviles tomando en cuenta la ubicación física de los mismos. Como ejemplo, tomemos el caso de una persona que llega al altar, en esta zona podríamos mostrar la información del altar proveyendo links a los autores de los detalles del mismo. Al cambiar la ubicación física del usuario la aplicación navegaría a otro nodo que representa la versión hipermedial de la zona. Debido a que estas aplicaciones hacen uso de conceptos de hipermedia en conjunto con un ambiente físico se las llama aplicaciones de Hipermedia Física [16, 17, 18].

Como hemos podido observar, estas aplicaciones se apoyan en conceptos más básicos como el clima, el tiempo y en particular la ubicación de los objetos. La ubicación ha sido uno de los primeros conceptos utilizados debido a que nos

brinda información útil a la hora de tomar decisiones. Si conocemos la ubicación de un objeto podemos estimar a qué distancia nos encontramos del mismo, qué objetos están cercanos y cómo hacemos para llegar hasta ellos. Es decir, que camino físico tenemos que seguir para que en un tiempo finito nos encontremos sobre ellos. Si tomamos en cuenta que un usuario puede moverse por diversos lugares como calles, autopistas, edificios, aviones, barcos y pasillos, el determinar la ubicación de un usuario no resulta una tarea sencilla. Aún cuando dispongamos de sensores esparcidos geográficamente y de los dispositivos de entrada salida necesarios, la cantidad de información y relaciones físicas existentes entre los objetos se tornará inmanejable. Para atacar este problema, debemos abstraer cuales son las partes importantes del mismo para luego producir un modelo que de solución a este problema. Como el modelo no es la realidad, éste nunca será completo y habrá que rediseñarlo para que se adapte a los cambios. Este trabajo esta motivado por la existencia de varios modelos que permiten trabajar con ubicaciones (Capítulo 2 y 3) los cuales han tratado de atacar el problema desde diferentes puntos de vista sin resultados destacados. Todos ellos se han limitado a tratar a la ubicación como algo que puede ser modelado en su totalidad olvidándose lo que se está intentando realizar, una vista parcial de la realidad. Tampoco se ha tomado en cuenta la idea de componer los modelos con el fin de obtener una visión más completa del mundo físico. Si tuviéramos un modelo de las ubicaciones de La Catedral y otro modelo de las calles de la ciudad, querríamos indicar que calle es la que pasa por la entrada de la misma. Esto permitiría que cualquier programador utilice los modelos para desarrollar aplicaciones móviles que permitan, por ejemplo, calcular un camino desde una calle de la ciudad hasta el altar. También permitiría enriquecer el modelo original cuando este es integrado con nuevos modelos, como podría ser el modelo de ubicaciones de un centro cultural. Por ser este uno de los problemas a resolver por las aplicaciones móviles, se espera que este trabajo provea una arquitectura que permita desarrollar modelos de ubicaciones con una visión constructiva y que a su vez permita integrar de una manera sencilla los diferentes modelos de ubicación.

1.2. Objetivo

Los modelos de ubicación se desarrollan con el fin de expresar las relaciones físicas entre los objetos. Como todo modelo, puede ocurrir que surjan nuevos requerimientos luego de que este fue puesto en producción y que estos provoquen modificaciones en el modelo original. Dependiendo de cuán correcto y con que fin fue diseñado, estos cambios pueden provocar que el modelo quede obsoleto y se tenga que rediseñar.

Los modelos de ubicación existentes (Capítulo 2 y 3) son adecuados depen-

diendo del tipo de aplicación que estemos desarrollando. Por lo tanto debemos conocer con anterioridad cuales son los requerimientos que esta posee para tomar la decisión de cual de ellos elegir. Si tomamos en cuenta que con la aparición de los dispositivos móviles los requerimientos pueden ser aún más cambiantes y que los modelos de ubicaciones podrán ser utilizados por múltiples aplicaciones, es probable que en un tiempo menor al esperado nuestro modelo quede obsoleto. Los modelos actuales intentan modelar a la ubicación como algo formalizable lo cual provoca que los mecanismos para agregar semántica a las ubicaciones no sean tomados en cuenta. Por este motivo, el primer objetivo que tiene este trabajo es el de proveer un modelo que permita enriquecer semánticamente a las ubicaciones de manera que los problemas de escalabilidad encontrados en las implementaciones actuales desaparezcan. De esta manera se permitirá que un modelo evolucione agregando semántica mediante distintas representaciones de una misma ubicación.

Si pensamos en un ambiente donde las ciudades, los edificios, las rutas y autopistas publiquen sus modelos de ubicación, podríamos desarrollar aplicaciones que independientemente del lugar donde nos encontremos, nos permitan estar ubicados siempre. Los modelos que han sido desarrollados hasta el día de hoy, sólo han sido pensados con el objetivo de satisfacer los requerimientos de una aplicación en una zona geográfica en particular. Los mismos no han sido pensados con el fin de ser compuestos permitiendo generar un modelo más grande en el sentido de cantidad de ubicaciones conocidas. Es por ello que este trabajo tiene como segundo objetivo permitir una manera sencilla de componer los modelos.

De la misma forma en que las personas contribuyen en Wikipedia[1] aportando sus conocimientos y en Google Earth[2] aportando los lugares donde se encuentran sitios de interés, cualquier persona que no posee los conocimientos tecnológicos suficientes podría contribuir generando nuevos modelos y componiéndolos con los existentes. Por este motivo se esperan dar un conjunto de guías con el fin de que una persona sin conocimientos técnicos pueda realizar dicha tarea.

1.3. Estructura del trabajo

Para abordar el trabajo con ubicaciones será necesario presentar un conjunto de conceptos básicos y de conocer los modelos básicos existentes (Capítulo 2). Para ello se explicarán conceptos como: objeto ubicable, ubicación, sistema de ubicación y mapa.

Luego, se desarrollará el estado del arte (Capítulo 3) de sistemas de ubicación. Para ello se detallará como un conjunto de aplicaciones han utilizado

[1] http://www.wikipedia.org/
[2] http://earth.google.com/

mecanismos *ad-hoc* para desarrollar sistemas de ubicación y cómo distintas arquitecturas y frameworks intentan resolver los problemas anteriormente enunciados.

Entendiendo los conceptos (Capítulo 4) podremos comprender por qué los trabajos anteriores atacaban el problema desde un punto de vista conceptual incorrecto. Para ello se expondrán conceptos como representación y sistema de representación los cuales provocarán una replanteo de los conceptos anteriores.

Sobre estos últimos, los conocimientos básicos sobre ubicaciones y la toma de conocimiento de los desarrollos que existen en la actualidad, se presentará la arquitectura (Capítulo 5) que espera resolver los problemas enunciados en el objetivo del presente trabajo. Se mostrará como se ha instanciado dicha arquitectura en un framework que permite resolver los problemas de escalabilidad e integración enunciados anteriormente.

Con el fin de validar el framework desarrollado, se mostrará como se ha desarrollado un caso de prueba (Capítulo 6). Dicha aplicación permite computar recorridos que involucren diferentes modelos de ubicaciones en tiempos razonables de cómputo.

Con el fin de que cualquier persona pueda participar en la construcción de un modelo se darán guías de diseño (Capítulo 7). Estas guías servirán para que una persona pueda modelar objetos ubicables y establecer relaciones físicas entre los mismos, sin tener los conocimientos técnicos para conocer la arquitectura y el framework.

Por último, se presentará las conclusiones del trabajo realizado y se enunciarán algunos temas que han quedado fuera del mismo (Capítulo 8).

Capítulo 2

Conceptos de ubicación

2.1. Conceptos

La ubicación es uno de los aspectos sobre los cuales distintas áreas dentro de las ciencias de la computación se han apoyado para desarrollar sus trabajos. Éstas han desarrollado distintos modelos de ubicaciones, pero al ser éste sólo un punto de apoyo, la forma en la cuál era modelada no era lo importante. Lo que si era importante era cual era su semántica y la semántica de sus operaciones. Conocerlas permite trabajar con ellas independientemente de como se encuentran implementadas. Para conocerlas es necesario entender el significado conceptual de términos como: ubicación, objeto ubicable y sistema de ubicación entre otros.

2.1.1. Ubicación

Las personas utilizamos constantemente e inconscientemente el termino ubicación. Por ejemplo, cuando un policía de un lugar turístico le explica a los turistas donde se encuentra una atracción de interés, le da una secuencia de lugares claves por los cuales tiene que pasar para no perderse en el camino. Esta secuencia de lugares tienen la particularidad de que sus ubicaciones se encuentran cercanas. En este caso, no se hizo un uso explícito del término y por ello su uso cotidiano es tan natural que muchas veces no nos damos cuenta que lo estamos utilizando y mucho menos nos cuenstionamos su significado. Simplemente nos alcanza con saber para qué nos resulta útil conocer una ubicación. Por ejemplo, si conocemos la ubicación de dos objetos que se encuentran adyacentes entonces sabemos que los objetos están próximos en un sentido físico.

Una de las formas que resulta más interesante de explicar que significa un término es conociendo sus objetivos y/o propiedades. Por ejemplo una propie-

dad de una ubicación es que **si** conocemos la ubicación de los objetos **entonces** podremos determinar cuan cercanos se encuentran. Sin embargo, existen diferentes definiciones que intentan formalizar el significado del término. La Real Academia Española [11] define el término ubicación como la acción y efecto de ubicar. Wikipedia [40] lo define como la posición en el espacio físico expresada en forma relativa a la posición de otro punto o cosa. Es decir, la ubicación de un objeto es expresada en términos de la ubicación de otros objetos. Si siempre debemos expresar la ubicación en términos de otra, ¿cuando se termina el proceso?. Por este motivo se habla de ubicación absoluta cuando no se necesita nada más que la ubicación para expresarla. Por el contrario, hablamos de ubicaciones relativas cuando expresamos una ubicación en términos de otras. Supongamos que la ubicación de la Casa de Gobierno es conocida en forma absoluta, por tal motivo decir que una persona se encuentra ubicada en ella es dar la ubicación de la persona en términos absolutos. En cambio, si le solicitamos a un persona que nos alcance el martillo explicándole que el martillo se encuentra sobre la mesa roja, estaremos expresando la ubicación del martillo en forma relativa con respecto a la ubicación de la mesa roja.

La definición propuesta en Wikipedia nos habla del término posición en relación con el concepto de ubicación, por lo cual es conveniente mencionar que una posición no es lo mismo que una ubicación. Hablamos de posición cuando nos referimos a una figura geométrica, por ejemplo un punto en un plano, la cual puede ser utilizada como forma de modelar una ubicación. Es decir, una posición puede ser una forma de modelar la ubicación de un objeto en un espacio geométrico. Como veremos más adelante (Sección 2.2 y Capítulo 3) también existen otras formas de modelar una ubicación.

En conclusión, las definiciones encontradas no parecen converger en algo en común, por lo tanto es conveniente retomar la idea de explicar que es una ubicación en términos de que puedo hacer con ella. De esta manera podremos ver a una ubicación como un ente abstracto el cual queda conceptualizado a partir de los requerimientos que puede satisfacer. En forma intuitiva podemos ver que si conocemos un conjunto de ubicaciones podremos establecer distancias, relaciones de inclusión y adyacencia entre los elementos de dicho conjunto. Seguramente nos gustaría que una ubicación nos permita establecer otros tipos de relaciones físicas existentes. Sin embargo cubrir la totalidad de las mismas sería bastante complejo. Por este motivo y en base a un análisis de la literatura [25, 39, 34, 31, 27, 28] se han encontrado que las operaciones más utilizadas son: inclusión, distancia, adyacencia y cómputo de caminos. En este sentido debemos determinar qué operaciones se pueden establecer en términos de otras. De esta manera acotaremos el conjunto de operaciones primitivas y podremos establecer el resto en términos de éstas. Por ejemplo, el cómputo de caminos puede realizarse con las operaciones de adyacencia y distancia por lo cual no es considerado una primitiva. El resto de las operaciones, al no poder ser expresadas en

términos de otros, son consideradas primitivas que una ubicación debe entender.

La idea de entender qué es una ubicación en términos de qué cosas puedo hacer con ella es interesante. Para comprender que puedo hacer con ella será necesario conocer la semántica de cada una de las operaciones. De esta forma podemos trabajar con ubicaciones independientemente de como se encuentren definidas a través de sus operaciones.

La operación de inclusión, generalmente simbolizada $\subseteq$, posee la semántica asociada a la formula proposicional: $\alpha \subseteq \beta$ sii $\forall x \in \alpha \rightarrow x \in \beta$. Es decir, la ubicación A contiene a la ubicación B si todo objeto que está en B está en A. Por ejemplo, las ubicaciones de la Catedral de La Plata (B) están incluidas dentro de la ubicación de la Ciudad de La Plata (A). Formalmente, la operación de inclusión debe satisfacer las siguientes propiedades:

- Reflexividad: $\forall \alpha \subseteq \alpha$
- Antisimetría: $\alpha \subseteq \beta \wedge \beta \subseteq \alpha \rightarrow \alpha = \beta$
- Transitividad: $\alpha \subseteq \beta \wedge \beta \subseteq \eta \rightarrow \alpha \subseteq \eta$

Muchas veces utilizamos la inclusión como si se satisfaciera la condición algebraica de supremo [30] . Es decir, para todo par de ubicaciones A, B existe un **único** objeto C que contiene a A y B y cualquier otro objeto que contenga a A y B llamémoslo D, C $\subseteq$ D $\wedge$ C $\neq$ D. Aunque esto puede llegar a ser correcto, desde el punto de vista de como las personas construyen sus modelos puede que no sea así.

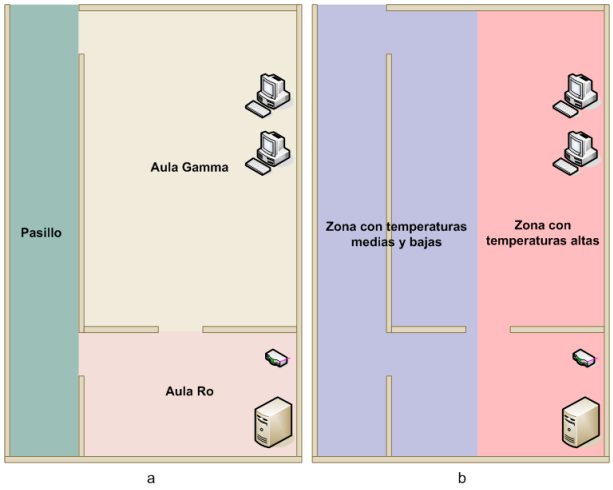


Figura 2.1: Operación de inclusión y supremo

Supongamos que una persona define un modelo en donde dos computadoras se encuentran incluidas físicamente en una sala (Figura 2.1 a). Por otro lado, otra

persona afirma que ambas computadoras se encuentran incluidas en la zona de temperaturas altas del edificio (Figura 2.1 b). Ambas afirmaciones son correctas, pero no se da el caso en que ni la zona contenga por completo a la sala, ni la sala contenga por completo a la zona. Por lo tanto no podemos asegurar que exista un único objeto que incluya a ambas computadoras y sea mínimo. Por este motivo la operación de inclusión desde un punto de vista de modelos de ubicación no asegura la propiedad de supremo.

La adyacencia es un concepto que aún no posee una definición formal. En general, las personas utilizan el término para expresar el hecho de que dos objetos se tocan o para indicar que a partir de un objeto puedo alcanzar otro. Por ejemplo, la bicicleta es adyacente a la puerta ya que **si** llego hasta la bicicleta **entonces** estoy muy cercano de alcanzar la puerta. En general ocurre que si nos encontramos en una determinada ubicación podemos pasar a una adyacente y viceversa. En tal caso decimos que la operación de adyacencia A satisface la propiedad de simetría, es decir, $\alpha A \beta \rightarrow \beta A \alpha$. Sin embargo, existen casos en donde la adyacencia no satisface esta propiedad. Supongamos que nos encontramos en un comedor universitario. La zona donde se sirve la comida (Figura 2.2) esta organizada de manera tal que la entrega sea realizada eficientemente. Para ello se han determinado tres zonas, la primer zona es donde los estudiantes se sirve la entrada, la segunda zona es donde se sirve el plato principal y en la última zona es donde se sirve el postre. Los alumnos recorren las zonas en un sentido incremental desde la entrada, eligiendo el plato principal y luego el postre. Si expresamos las relaciones que existen entre las ubicaciones de cada una de las zonas, seguramente diremos que la ubicación donde se elige la entrada nos permite pasar a la ubicación donde se sirve el plato principal, pero no viceversa. Análogamente ocurre con la ubicación donde se sirve el plato principal y la ubicación donde se sirve el postre.

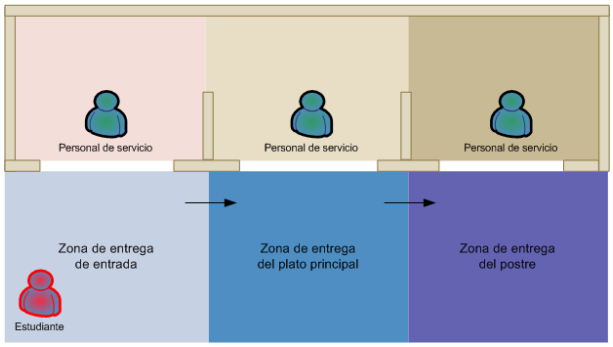


Figura 2.2: Adyacencia y no simetría

La distancia (simbolizada $\odot$), en términos físicos, habla de una medida de cuan cercanos se encuentran dos objetos. Es uno de los requerimientos más

conocidos entre las personas debido a que su uso es aun más cotidiano que el resto. Por ejemplo, utilizamos la distancia para referirnos a cuan cercanos están la ciudad de La Plata y la Ciudad Autónoma de Buenos Aires cuando decimos que se encuentran a 60 km. La distancia física tiene sus orígenes en la geometría, la cual establece un conjunto de propiedades que tiene que satisfacer:

- No negatividad: $\alpha \odot \beta \geq 0$
- Simetricidad: $\alpha \odot \beta = \beta \odot \alpha$
- Desigualdad triangular: $\alpha \odot \beta \leq \alpha \odot \eta + \eta \odot \beta$

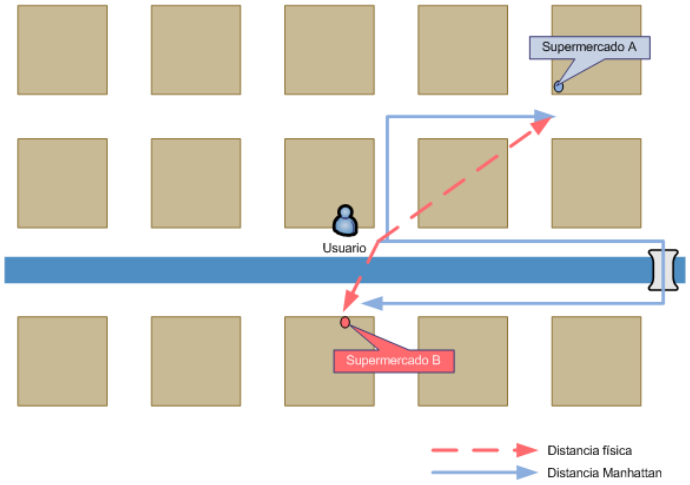


Figura 2.3: Distancia física vs. distancia manhattan

El término distancia es utilizado también con otras connotaciones. Por ejemplo, es común que deseemos calcular la distancia entre dos edificios de una ciudad. Si utilizamos la noción de distancia anteriormente citada, la misma será calculada como si pudiéramos recorrer el espacio el línea recta. Es claro que esto no ocurre en una ciudad, ya que debemos sortear las calles y edificios. Por este motivo, se utiliza el concepto de distancia de Manhattan para referirse a la distancia que debemos calcular en base a un recorrido, el cual puede no ser realizado en línea recta. Para ver la diferencia entre ambos conceptos, supongamos que queremos buscar el supermercado más cercano a la ubicación del usuario (Figura 2.3). Si optamos por el más cercano en cuanto a la distancia física eligiríamos el que se encuentra cruzando el río. Si utilizáramos el concepto de distancia de Manhattan eligiríamos el supermercado que se encuentra a 3 cuadras de la ubicación actual del usuario. Debemos notar que el recorrido real que el usuario tendría que realizar si optáramos por la distancia física sería

mayor que el que realizaríamos si hubiéramos calculado el camino en base a la distancia de Manhattan. Es decir, la distancia de Manhattan nos permite dar una estimación real de distancias físicas en base a obstáculos que presenta el recorrido en línea recta.

2.1.2. Sistema de ubicaciones

Áreas como el Context Aware y la Hipermedia física utilizan a la ubicación para desarrollar sus objetivos. Lo interesante es que estas áreas no utilizan ubicaciones que se encuentran previamente cargadas en el sistema sino que utilizan ubicaciones que pueden ir cambiando a lo largo del tiempo. Es decir, utilizan la ubicación actual del usuario asumiendo que ésta es la misma que su dispositivo móvil. Por este motivo, se necesita de algún mecanismo que permita obtener información que pueda ser traducida a ubicaciones. Esta tarea es realizada mediante diferentes tecnologías de sensores que permiten, a través de señales eléctricas, infrarrojas o sonoras establecer con gran exactitud la ubicación del dispositivo. Esta transformación de señal recibida a ubicación no es directa, es necesario un proceso que abstraiga las señales de bajo nivel en elementos útiles para el desarrollador como son las ubicaciones. Al conjunto de sensores, transformaciones de señales y modelos de ubicaciones se acostumbra a llamarlos sistemas de ubicación [19].

Esta definición de sistema de ubicación muestra como existe un alto nivel de acoplamiento entre las ubicaciones y los mecanismos de sensado. Por este motivo, algunas publicaciones [34, 3] utilizaron luego el término sistema de soporte para ubicaciones, para referirse a los sistemas de hardware que dan soporte a la generación de ubicaciones en base a señales.

2.1.3. Objeto ubicable

Las ubicaciones no tendrían sentido de existir si no denotaran el contexto físico de un objeto. Lo que nos interesa es un objeto y su ubicación, por este motivo, Schilit introdujo el concepto de objeto ubicable [36] para referirse a aquellos objetos que poseen una descripción y se encuentran asociados a una ubicación. La descripción utilizada por Schilit es debido a que su trabajo trata a los objetos ubicables dentro del contexto de un servicio de mapas. Por ello, la descripción del objeto ubicable era lo que le permitía identificarlos. En la misma época Spreitzer [37] desarrolla una representación para objetos ubicables dentro de su área de trabajo, las redes. Dicha representación modelaba a un objeto ubicable como una tupla formada por una ubicación, un canal RPC y un tipo. El canal RPC muestra el alto acoplamiento que tenía su representación con el objeto de su trabajo. Sin embargo, es posible pensar en una representación más sencilla de objeto ubicable, en donde lo único diferente con respecto a otro tipos

de objetos es la ubicación. Es por ello que en base a los trabajos presentados por ambos, utilizaremos el término objeto ubicable como un objeto que posee una ubicación.

La definición de objeto ubicable no nos dice nada respecto de qué tipos de objetos pueden o no ser objetos ubicables, por lo tanto podemos asumir que cualquier objeto donde su ubicación sea un aspecto de interés puede ser considerado ubicable. Por ejemplo, una persona podría ser un objeto ubicable ya que podríamos decir que una persona esta incluida en una sala y que se encuentra adyacente a una computadora. Esta última, también es un objeto ubicable si decimos que se encuentra ubicada en el salón dorado y que pertenece a la red Alpha. Notemos que en este caso hemos utilizado el término ubicación con una connotación distinta a la utilizada hasta el momento. Nos hemos referido a la ubicación de una computadora en una red de computadoras, lo cual es independiente de donde esta se encuentre ubicada físicamente. Este hecho también ocurre cuando trasladamos nuestra notebook o Smartphone y accedemos a internet a través de diferentes redes WIFI. El dispositivo puede cambiar su ubicación física, por ejemplo si estamos en un bar con conexión WIFI y nos cambiamos de una mesa a otra, pero sin cambiar su ubicación virtual, su ubicación en la red. Debido a que el presente trabajo tiene como objetivo resolver problemáticas físicas, el término sólo será utilizado con dicha connotación y en caso que se desee expresarlo en otro contexto como puede ser el virtual, se realizará explícitamente.

2.1.4. Mapa

Cualquier objeto podría ser considerado un objeto ubicable, lo cual origina que en una aplicación existan numerosos objetos ubicables. Naturalmente, comprender cómo se relacionan en base a sus ubicaciones resulta una tarea difícil de llevar a cabo. Con el objetivo de comprender sólo un subconjunto del mismo es que se han construido los mapas. El concepto de mapa viene desde la época de los aborígenes que pintaban mapas para mostrar las ubicaciones de sus aldeas. Uno de los ejemplos más conocidos son los mapas que utilizaban los piratas para encontrar los tesoros (Figura 2.4).

Los mapas son elementos utilizados desde épocas antiguas con el objetivo de resaltar un conjunto de objetos. En general, los mapas tienen un sentido de existir. Por ejemplo, los mapas del encuentro del tesoro tienen como objetivo que las personas que los visualicen sean capaces de encontrar el tesoro. Los mapas que podemos encontrar en las estaciones de servicio tienen como objetivo ubicarnos en un contexto como puede ser una ciudad o un país. Por ello, no tendría sentido mostrar en el mapa de La Ciudad de La Plata los objetos ubicables que una persona tiene en un departamento ya que ellos no tienen que ver con el objetivo del mapa.

Figura 2.4: Mapa del tesoro

Desde un punto de vista de ingeniería de software un mapa podría ofrecer un conjunto de servicios interesantes. Por ejemplo, un mapa nos podría ayudar a encontrar objetos que satisfagan un conjunto de condiciones y brindar servicios de cómputos de camino entre dos o más ubicaciones.

2.2. Formas sencillas de modelar ubicaciones

A pesar de que no lo hagamos conscientemente, en nuestro día a día utilizamos constantemente la noción de ubicación; un ejemplo es cuando los turistas que llegan a una ciudad y solicitan a los policías cómo llegar a una atracción turística como puede ser la Catedral de La Plata. Debido a que sólo se encuentran a unas cuadras, el policía les indica que deben doblar a la derecha en la próxima esquina y luego caminar unos 200mts (Figura 2.5). Otro ejemplo del uso de la ubicación, es cuando el dueño de una tienda le indica a un empleado que le alcance la carpeta que tiene las órdenes de compra. Como el empleado no conoce su ubicación, el dueño le indica que la carpeta se encuentra en el tercer estante del mueble de madera, dentro de la oficina del jefe de compras.

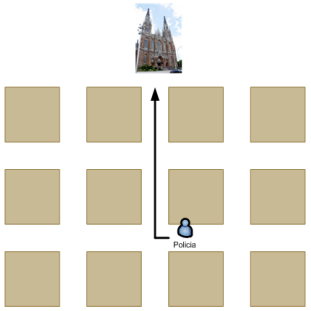


Figura 2.5: Policía indicando como llegar hasta la catedral

En estos ejemplos podemos ver que tanto el policía como el dueño de la

empresa utilizaron distintos modelos que les permitieron razonar sobre las ubicaciones. En el primer caso el policía utilizó un modelo basado en distancias y direcciones, el cual le permitió orientar a los turistas. Por otro lado, el dueño utilizo algo que generalmente es más sencillo de interpretar por el ser humano como son los nombres. Para ello el empleado debe tener un conocimiento previo de que el nombre oficina denota el cuarto que se encuentra al lado del pasillo y que el pasillo es otro nombre que denota el lugar que contiene una máquina de café. Notar que esto se asemeja a como el ser humano construye una red de conceptos en base a un conjunto de nombres y asociaciones que le permiten relacionarlos (Figura 2.6). En el caso de las ubicaciones, hay algunas relaciones que, como habíamos dicho anteriormente, surgen naturalmente como son la adyacencia, la distancia y la inclusión. De esta manera el empleado utiliza estas relaciones para deducir como llegar hasta la carpeta. Por ejemplo, asumamos que el empleado sabe que se encuentra en la sala de reuniones y que esta tiene adyacente una puerta, la cual es adyacente al pasillo. Esta última es adyacente a la oficina del jefe de compras la cual contiene un mueble de madera, el que contiene estantes, dentro de los cuales uno es adyacente a la carpeta buscada.

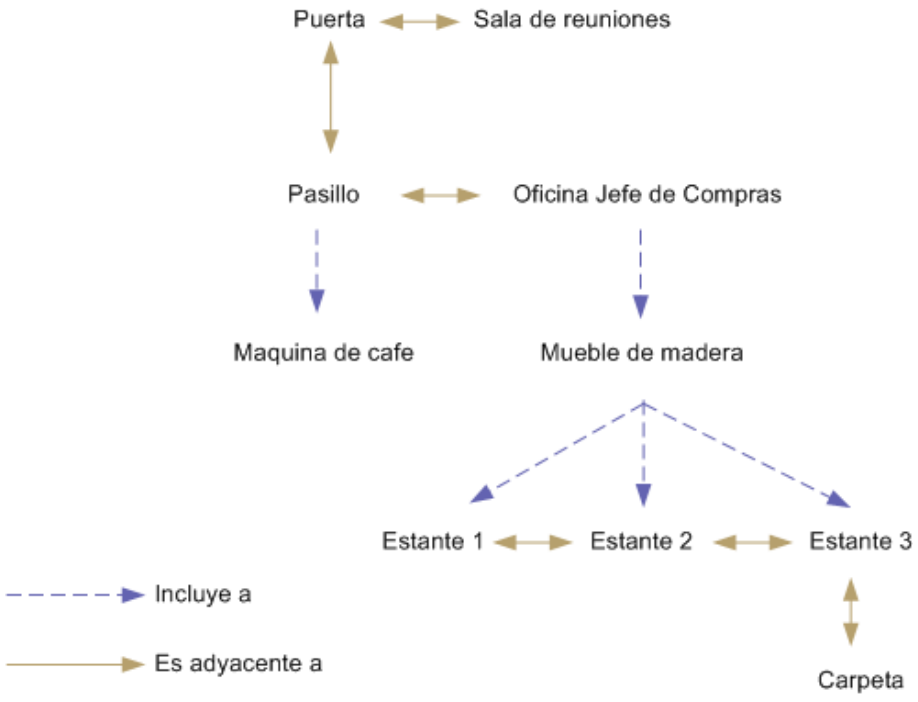


Figura 2.6: Red conceptual del dueño de la empresa

Los modelos utilizados por el policía y el jefe de compras son los que utilizamos comúnmente para modelar ubicaciones. Por ejemplo podríamos modelar

tanto geométricamente como simbólicamente el parque (Figura 2.7) pero la elección de cual de estos modelos utilizar, deberá realizarse tomando en cuenta las ventajas y desventajas que estos presentan.

Figura 2.7: Parque

2.2.1. Modelos geométricos

Los modelos geométricos están basados, como su nombre lo indica, en el concepto de geometría. La palabra geometría proviene del griego *geo*, que significa tierra y *metría* que significa medida. Es una rama de las matemáticas que se ocupa del estudio de las medidas, el tamaño y la posición relativa de figuras y sus propiedades en el espacio. Es este sentido una figura en un espacio geométrico representa la forma de modelar una ubicación. Dentro del conjunto de geometrías existentes, la más conocida y utilizada es la geometría euclidiana. La geometría euclidiana está basada en conjuntos finitos de tuplas en el espacio $\mathbb{R}^n$ con $n \in \mathbb{N}$. De esta manera si tenemos la geometría en el plano (n = 2), podemos definir la figura línea desde la dupla[1] (0,0) hasta la dupla (3,3), como el conjunto de duplas (x,y) que satisface la ecuacion: $y = x \wedge 0 <= x <= 3$. Los modelos geométricos utilizados para modelar ubicaciones son los planares (n = 2) y los espaciales (n = 3), aunque aquí sólo se comentarán los modelos planares debido a que son más simples que los espaciales.

Los modelos planares modelan a las figuras en base a tres figuras primitivas: el punto, la polilínea y el polígono. Con estas tres figuras se pueden especificar tres conjuntos de duplas diferentes. Aquellos que sólo un elemento del plano los satisface como es el punto. Aquellos que son satisfechos mediante una ecuación o una unión de ecuaciones las cuales poseen al menos un punto en común, como es una polilínea. Por último aquellos puntos que satisfacen un conjunto de inecuaciones como son los polígonos (Figura 2.8).

[1] Indistintamente nos referiremos a las duplas como puntos

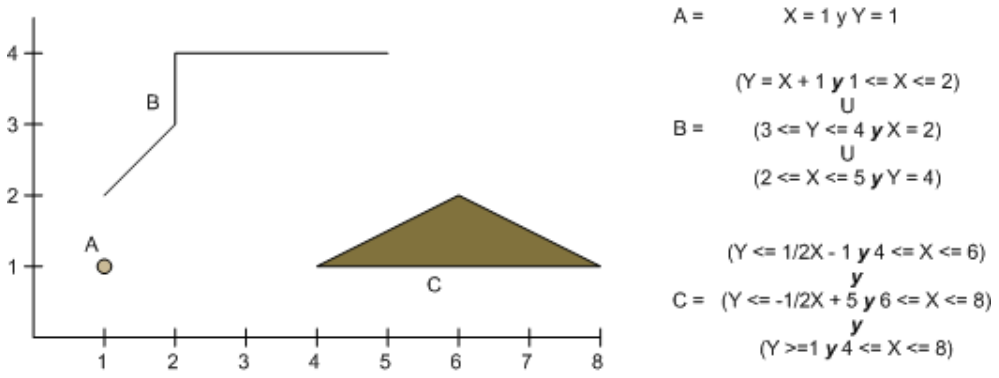


Figura 2.8: Punto, polilínea y polígono

Las figuras que modelan a las ubicaciones deben definir el conjunto de operaciones primitivas que habíamos identificado para las ubicaciones (Sección 2.1.1). Afortunadamente, el modelo geométrico, al estar basado en la geometría, posee una definición formal de las mismas las cuales le permite satisfacer las propiedades enunciadas. Siendo F_1 y F_2 figuras del plano y recordando que una figura representa un conjunto de duplas, las operaciones se encuentran definidas de la siguiente forma:

- Inclusion: $F_1 \subseteq F_2$ sii $\forall (x, y) \in F_1 \rightarrow (x, y) \in F_2$
- Distancia: $d(F_1, F_2) = min\{\ n \mid n = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\}$ con $(x_1, y_1) \in F_1$ y $(x_2, y_2) \in F_2$
- Adyacencia: $F_1 \approx F_2$ sii $F_1 \cap F_2 \neq \varnothing \wedge \forall (x, y) \in F_1 \cap F_2, (x, y) \in borde(F_1) \vee (x, y) \in borde(F_2)$ con:
  $borde(F) = \{(x, y) | (x, y) \notin abierto(F)\}$ y
  $abierto(F) = \{(x, y) | \exists r \in I\!R, r > 0, \forall (a, b) \in$ círculo con radio r y centro $(x, y) \rightarrow (a, b) \in F\}$

Los modelos planares pueden ser definidos en términos de otros modelos. Por ejemplo, podemos definir un modelo geométrico como una translación de otro. Es decir, todo punto del nuevo modelo puede ser transformado a un punto del modelo anterior mediante una función. La traslación no es la única función que se utiliza para realizar transformaciones. El escalado y la rotación son también dos casos de los más utilizados. La transformación puede ser simple, es decir utilizando una única función o ser una composición de funciones. Por ejemplo podemos definir un modelo como una traslación + rotación de otro (Figura 2.9).

Las figuras representan conjuntos finitos o infinitos de duplas, es por ello que cuando estos modelos son traducidos a una computadora debemos discretizarlos de alguna forma. Por este motivo existen diferentes estrategias a la

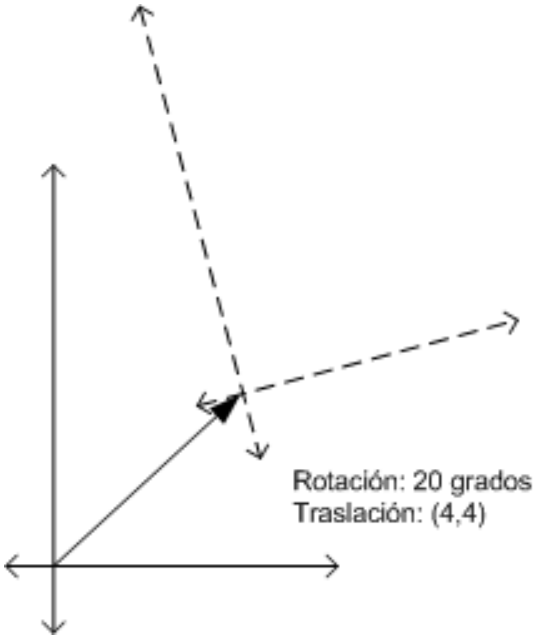


Figura 2.9: Rotación y traslación

hora de modelar dichas figuras computacionalmente. Algunas de ellas priorizan la simplicidad, otras permiten variar fácilmente el modelo a través del tiempo (sin demasiado costo computacional) y otras mejoran la performance de algunas operaciones.

Spaghetti

La forma más sencilla de modelar las tres figuras geométricas primitivas es mapeandolas directamente en el plano. Es decir, los puntos son modelados como una dupla (x, y), las polilíneas son modeladas como una secuencia de n duplas (x, y) en donde se asume que la polilínea esta expresada por la secuencia de segmentos de línea descriptas por los pares (x_i,y_i) (x_{i+1},y_{i+1}) con $i \in 1..n-1$, que representan el comienzo y fin del segmento. Por último los polígonos son modelados como una secuencia cerrada de n pares (x, y) que definen el borde del polígono. Éste queda determinado por todos los puntos (a, b) que satisfacen las inecuaciones provistas por cada uno de los segmentos de linea que forman el borde. Esta forma de modelar las tres figuras geométricas es la realizada por el modelo spaghetti.

Volviendo al ejemplo del parque, podemos utilizar este modelo para construir una representacion geométrica del mismo indicando una figura para la ubicación de cada uno de los objetos. En este caso hemos asumido que la zona de los pinos

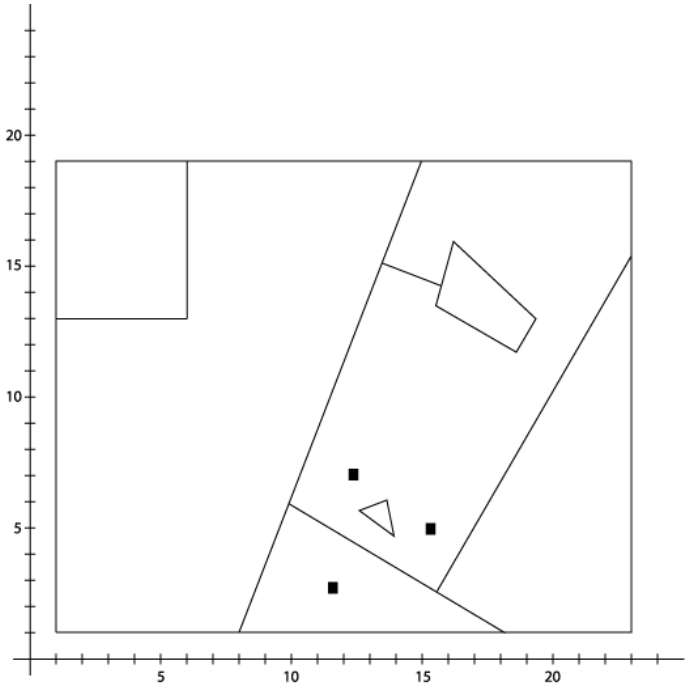


Figura 2.10: Modelo spaghetti del parque

y las mesas eran de poca importancia y que podían ser modeladas como un polígono en el primer caso y como puntos en el segundo (Figura 2.10). Los caminos fueron modelados como una polilínea debido a que éste se extiende de norte a sur y también algunos de ellos con dirección oeste. Por su parte el lago y el estacionamiento fueron modelados como un polígono que representa el área cubierta por cada uno de ellos sobre la superficie terrestre.

Raster

El modelo geométrico raster consiste en una cuadrícula **regular** de celdas, en donde las figuras geométricas son discretizadas, pudiendo ocupar más de una celda. La forma de las celdas es cuadrada o rectangular, aunque también existen versiones triangulares.

La versión original del modelo raster permite asignar un valor único a cada celda. De esta manera no podríamos tener dos figuras geométricas que ocupen una misma celda. Para permitir este hecho se introduce el concepto de capa o layer. Una capa representa una vista sobre las celdas, permitiendo representar diferentes figuras sobre una misma celda.

A diferencia del modelo spaghetti donde podíamos establecer puntos en forma arbitraria, cada celda del raster representa un área de la superficie real. Es

decir, el plano es discretizado en áreas de forma regular que cuanto menor es el tamaño del área, mayor será su resolución. Con resolución nos referimos a que el modelo asemeje las figuras ideales. Que la resolución sea alta implica que necesitaremos mayor cantidad de espacio para almacenar las celdas.

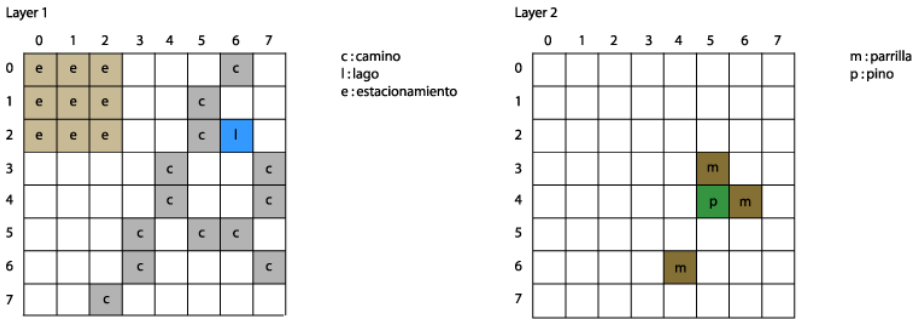


Figura 2.11: Modelo raster del parque

Volviendo al ejemplo del parque, si discretizamos su imagen y coloreamos cada celda se podrá observar que el modelo raster se asemeja a una imagen pixelada (Figura 2.11). El lago y el estacionamiento quedan modelados como un conjunto de puntos adyacentes, los pinos y las mesas dependiendo del nivel de resolución del raster, pueden desaparecer del modelo o ser modelados como una celda dentro del raster. Por último los caminos quedan modelados como una secuencia de celdas adyacentes. Esta es una de las posibles formas de modelar las ubicaciones de los objetos del parque. Podríamos haber optado por eliminar las mesas debido a que la resolución del raster no es buena, por lo cual no es necesario ver las mesas en ese nivel de detalle. La decisión de qué modelar queda siempre para la persona que construya el modelo.

Topológico

El modelo topológico, como su nombre lo indica, mantiene la topología de las figuras. Para ello las figuras poseen información de cuales son las figuras con las cuáles están relacionadas. El modelo utiliza como unidad básica de modelado el arco, el cual representa una serie de puntos que comienza y termina en un nodo. Un nodo es un punto que intersecta a dos o más arcos. Un polígono se especifica mediante una cadena cerrada de arcos que representan los límites del área de la figura. Por último, si consideramos que el modelo está acotado, el área que se encuentra fuera de los límites del área se designa por medio de un polígono que no posee arco. Lo cual, desde un punto de vista algebraico, quedaría definido por: $\mathbb{R}^2 - \bigcup figura_i$ siendo $\bigcup figura_i$ la unión de todas las figuras del modelo. Para mantener la información de la topología de las figuras, el modelo utiliza relaciones de conocimiento. Por ejemplo, los nodos saben a qué arcos pertenecen,

es decir de qué arcos son extremos y los arcos conocen en qué nodos inciden y a que polígonos pertenecen.

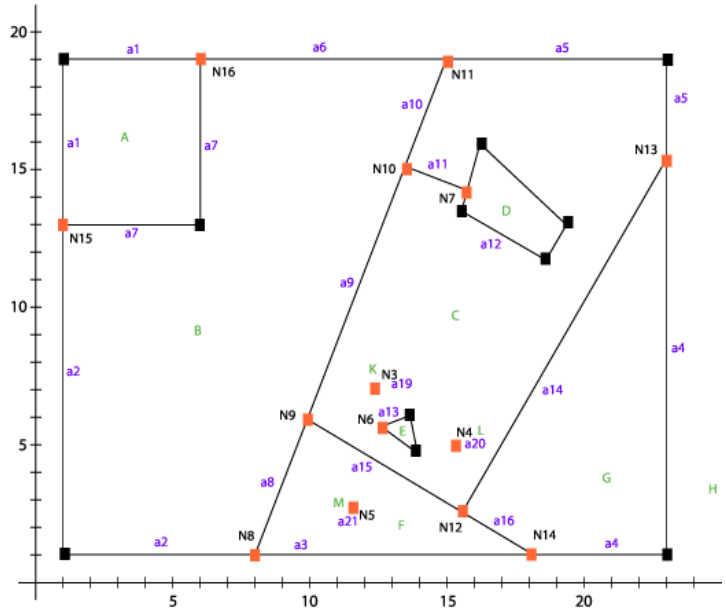


Figura 2.12: Modelo topologico del parque

El parque modelado con un topológico (Figura 2.12) resulta similar a un spaghetti. Se han diferenciado los nodos (con un color naranja) de los puntos intermedios de las figuras. El resto de los objetos ubicables resultan similares a como fueron modelados con un modelo spaghetti, aunque a diferencia de este, las relaciones topológicas fueron computadas y almacenadas en cada una de las figuras.

Ventajas y desventajas

Cada una de las implementaciones del modelo geométrico posee ventajas y desventajas. Dependiendo de los requerimientos que posea nuestra aplicación es cual de ellos eligiremos. Por ejemplo, el modelo **Raster** es sencillo y las operaciones son fáciles de implementar mediante operaciones de conjuntos entre las celdas. Sin embargo, las relaciones topológicas (como la relación de inclusión o adyacencia) entre las figuras son difíciles de representar y muchas veces el tamaño del raster ocupa demasiada memoria como para poder ser utilizado. El modelo **Spaghetti** resuelve estos problemas de memoria utilizando una estructura más compacta y permitiendo analizar las relaciones topológicas de una forma más sencilla. Sin embargo, en aplicaciones que hacen un fuerte uso de las operaciones y donde las figuras no cambian mucho, obligan al modelo a realizar

cómputo que podría haber sido resuelto con un modelo **Topológico**. El modelo Topológico utiliza una cache de las relaciones topológicas existentes entre las figuras para mejorar la performance de las operaciones. Sin embargo, si el modelo es dinámico en el sentido de que se agregan y quitan figuras del modelo con cierta frecuencia, el modelo se vuelve inadecuado ya que se desperdicia mucho tiempo recomputando dichas relaciones.

En base a este análisis, el modelo **Raster** resulta conveniente cuando lo que se quiere representar tiene carácter continuo y cuando se está dispuesto a pagar el precio de almacenamiento en base a algún beneficio en la discretización de las celdas. En cualquier caso se puede mejorar la calidad del modelo aumentando la resolución, aunque esto puede traer problemas de almacenamiento.

En contraposición al modelo Raster, el modelo **Topológico** es conveniente cuando se quiere representar de forma eficiente relaciones existentes entre objetos, como así también cuando se quiere representar objetos que se encuentran conectados. No resulta conveniente cuando las ubicaciones poseen mucha variabilidad ya que el costo de construir el modelo es alto.

Finalmente, el modelo **Spaghetti** resulta conveniente cuando no se requiere del almacenamiento de las relaciones entre los objetos. Esto podría ocurrir cuando lo que se quiere representar es muy variable respecto del intervalo de tiempo durante el cual se consulta. Por ejemplo, si consultamos el modelo cada 10 segundos y las figuras cambian cada 1, resulta adecuado ya que en caso de utilizar un modelo topológico realizaríamos más cómputo del necesario.

2.2.2. Modelos simbólicos

El modelo conceptual que la persona realizó en su mente se asemeja bastante a un modelo simbólico. Los símbolos como son pasillo, mueble, carpeta y las relaciones que existen entre ellos son lo que definen un modelo simbólico. Es decir para definir un modelo simbólico necesitamos de un conjunto de símbolos y un conjunto de relaciones existentes entre los símbolos. Por ejemplo, si asumimos que la flecha de color azul define la relación de inclusión entre los símbolos, entonces diremos que el símbolo a incluye al símbolo b en el caso que exista una flecha azul desde el símbolo a hasta el símbolo b.

La semántica de las operaciones primitivas se realiza definiendo tres relaciones sobre el conjunto de símbolos. Cada una de estas relaciones puede ser expresada de dos formas diferentes, por comprensión o por extensión. En el caso que definamos las relaciones por extensión, deberemos definir el conjunto de flechas en forma manual, esto resulta muy laborioso para modelos grandes. En cambio, si la semántica la definimos por comprensión, deberemos definir una regla que nos permita saber si dos símbolos se encuentran relacionados por la operación.

Supongamos que tenemos un modelo simbólico del parque nacional en don-

de las operaciones son definidas por extensión (Figura 2.13). Como se puede observar hemos establecido que el lago es adyacente al camino y que ambos se encontraban incluidos en el parque. Por otra parte, la distancia entre los pinos y las mesas es de 10 mts. En este caso, hemos omitido el estacionamiento por no ser un aspecto importante de la realidad. Un modelo similar podría haberse construido definiendo las relaciones por comprensión. La relación de inclusión se puede especificar con la regla que todos los símbolos están incluidos en el símbolo #parque. La distancia es 0 si dos símbolos son adyacentes y 1 caso contrario y por último la relación de adyacencia sería especificada por comprensión como en el modelo previo, debido a que dar una regla en este caso no resulta sencillo.

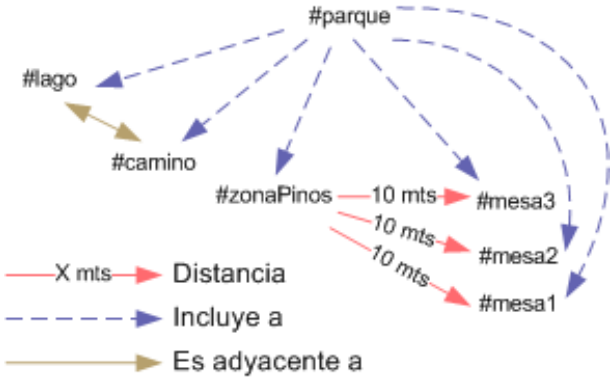


Figura 2.13: Parque simbolico

El modelo simbólico es confundido muchas veces con una red semántica [41, 32]. Existe una diferencia substancial entre una red semántica y un modelo simbólico. En una red semántica se abstraen un conjunto de conceptos como pueden ser: la zona de los pinos, las parrillas, el lago, el camino y en base a estos conceptos se definen un conjunto de relaciones. Por su parte un modelo simbólico también define un conjunto de elementos que se relacionan, pero estos elementos que se relacionan, los símbolos, pueden o no tener una semántica asociada. Por ejemplo, podríamos transformar el modelo del parque reemplazando los símbolos anteriores por símbolos aleatorios (Figura 2.14). Nada nos dice cada símbolo respecto a su semántica, pero si nos habla de las relaciones que este símbolo posee con otro. Las ubicaciones de cada uno de los objetos, dadas por las relaciones que tienen los símbolos, no se veerían alteradas ya que sólo se han reemplazado nombres. Por este motivo la diferencia entre una red semántica y un modelo simbólico es que la red semántica utiliza conceptos que poseen semántica, mientras que un modelo simbólico utiliza símbolos que pueden o no poseerla.

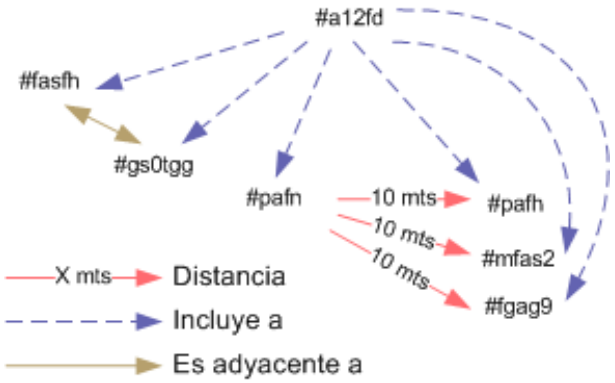


Figura 2.14: Parque simbólico aleatorio

2.2.3. Ventajas y desventajas

Tanto los modelos geométricos como simbólicos, son sencillos y relativamente fáciles de configurar, pero cúal de ellos utilizar y en base a que criterio, no resulta una tarea sencilla de resolver. Ulf Leonhardt en su tesis Doctoral [25] enumera un conjunto de ventajas y desventajas que poseen cada uno de ellos.

Los modelos simbólicos son más fáciles de utilizar que los modelos geométricos debido a que las ubicaciones pueden ser llamadas por un nombre. Esto nos permite controlar la seguridad de las ubicaciones de una manera más sencilla ya que es más sencillo saber a que ubicación nos estamos refiriendo por un nombre que por una secuencia de puntos. Por su parte el modelo presenta la desventaja de que los símbolos tienen que ser manejados en forma manual, es decir la creación de cada uno de los símbolos y su asociación con los objetos ubicables es una tarea que se establece manualmente.

Los modelos geométricos poseen la ventaja de ser más exactos si el modelo se ha construido realizando mediciones correctas, permitiendo que las operaciones entre las figuras sean precisas. A diferencia de los modelos simbólicos, donde debemos definir las relaciones, los modelos geométricos pueden ser utilizados sin tener que definir más que las figuras geométricas ya que poseen una semántica formal asociada. Por otro lado, presentan la desventaja de que se necesita algo más que información geométrica para establecer la seguridad de las ubicaciones, ya que un conjunto de puntos no nos dice nada respecto de a que objeto ubicable pertenece. Por último, si necesitamos transformar la información geométrica en información útil para los usuarios necesitaremos de un servicio adicional que nos permita transformar figuras en algo de mayor nivel de abstracción como un objeto ubicable.

Ambos modelos presentan sus ventajas y desventajas. En caso de necesitar un modelo simple, donde no tengamos muchos objetos ubicables (< 40) y donde

la precisión de las operaciones no es algo que sea importante, entonces será conveniente utilizar un modelo simbólico. En cambio si necesitamos desarrollar una aplicación en donde necesitemos precisión en las operaciones utilizaremos un modelo geométrico. Visto y considerando que cada uno de los modelos posee sus ventajas y desventajas, no se puede concluir que uno sea mejor que otro, pero si que uno resulta más adecuado dependiendo el tipo de aplicación que estemos desarrollando. La utilización de un modelo inicial no nos prohíbe que en un futuro necesitemos agregar otro modelo para aprovechar sus ventajas. De esta manera combinaríamos los modelos con el fin de obtener lo mejor de cada uno de ellos. Complementándolos se obtendría las ventajas de ambos, suavizando sus desventajas.

Capítulo 3

Estado del arte

3.1. Aplicaciones y arquitecturas

Los modelos geométricos y simbólicos, son dos formas sencillas de modelar ubicaciones. Como vimos en el Capítulo 2 las posibilidades que estos modelos presentan son acotadas. Esto provoca que sea necesario desarrollar nuevos modelos con el fin de mejorar los anteriores. Varias aplicaciones, arquitecturas y frameworks existentes, han ajustado o definido nuevos modelos con el fin de resolver distintos tipos de problemas. Conocer las aplicaciones nos permite observar implementaciones ad-hoc, sin generalizaciones, pero con un conjunto grande de casos de uso que se debe satisfacer. Esto nos permite conocer los requerimientos que debe satisfacer una arquitectura de sistemas de ubicación y también informarnos respecto de como se han resuelto problemas particulares. Por otro lado, las arquitecturas plantean soluciones a problemas más generales sin detallar en aspectos implementativos. Para validar el planteo teórico propuesto por la arquitectura, esta es instanciada en un framework que permite verificar su validez. Podemos observar que numerosas arquitecturas y frameworks extienden o mezclan los modelos con el fin de satisfacer nuevos requerimientos. Conocer las arquitecturas y frameworks existentes nos permite conocer como se resuelven algunos problemas y al mismo tiempo realizar una crítica constructiva sobre las mismas verificando si dichas soluciónes alcanzan para satisfacer los problemas enunciados en la introducción del presente trabajo (Capítulo 1).

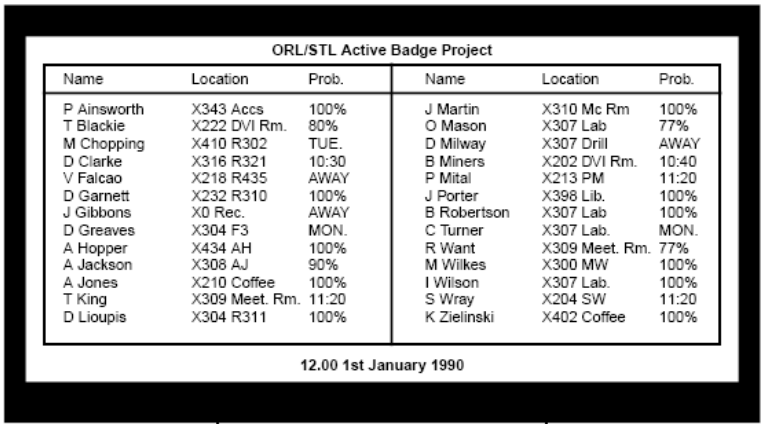
3.2. Aplicaciones

Las aplicaciones han sido las primeras formás de explorar un dominio desconocido. Los procesos de abstracción son generados desde los casos más básicos y en base a estos se pueden producir abstracciones que puedan ser reutilizadas.

Es por ello que presentaremos un conjunto representativo de aplicaciones que utilizan ubicaciones con el fin de mostrar como cada una de ellas las utiliza.

3.2.1. The Active Badge Location System

El Active Badge [39] es un sistema de ubicaciones que permite ubicar objetos dentro de ambientes cerrados. Aunque el sistema puede ser utilizado con fines más generales, uno de los usos que se le dio fue el de conocer la ubicación de cada empleado mostrándola en una pizarra electrónica (Figura 3.1). Esta aplicación es de extrema utilidad para una recepcionista ya que muchas veces necesita transferir llamadas y las personas no se encuentra ubicadas en sus puestos de trabajo. De esta forma, la recepcionista puede visualizar la posible ubicación de la persona redirigiendo correctamente la llamada.



ORL/STL Active Badge Project

Name	Location	Prob.	Name	Location	Prob.
P Ainsworth	X343 Accs	100%	J Martin	X310 Mc Rm	100%
T Blackie	X222 DVI Rm.	80%	O Mason	X307 Lab	77%
M Chopping	X410 R302	TUE.	D Milway	X307 Drill	AWAY
D Clarke	X316 R321	10:30	B Miners	X202 DVI Rm.	10:40
V Falcao	X218 R435	AWAY	P Mital	X213 PM	11:20
D Garnett	X232 R310	100%	J Porter	X398 Lib.	100%
J Gibbons	X0 Rec.	AWAY	B Robertson	X307 Lab	100%
D Greaves	X304 F3	MON.	C Turner	X307 Lab.	MON.
A Hopper	X434 AH	100%	R Want	X309 Meet. Rm.	77%
A Jackson	X308 AJ	90%	M Wilkes	X300 MW	100%
A Jones	X210 Coffee	100%	I Wilson	X307 Lab.	100%
T King	X309 Meet. Rm.	11:20	S Wray	X204 SW	11:20
D Lioupis	X304 R311	100%	K Zielinski	X402 Coffee	100%

12.00 1st January 1990

Figura 3.1: Pizarra de la aplicación desarrollada con Active Badge

Con el fin de conocer la ubicación de cada persona se utiliza un prendedor que emite señales que son capturadas por una red de sensores. El prendedor es tan pequeño que puede ser colocado en el bolsillo de la camisa por lo cual su uso no incomoda a ninguno de los usuarios. Las señales infrarrojas son emitidas durante un período corto de tiempo con el fin de que las señales de varios prendedores no se solapen. La señal es capturada por una red de sensores que se encuentran conectadas a la red de computadores de la organización (Figura 3.2). Esto permite reutilizar las redes de computadoras para transmitir información de donde se encuentran ubicadas las personas. Una vez que la señal es recibida por la red, es transmitida a una workstation para su procesamiento. Cada prendedor emite un identificador único que es sensado por un único sensor en un instante de tiempo, lo cual permite conocer la ubicación de la persona. Dentro de la arquitectura, las ubicaciones no se encuentran modeladas ya que alcanza con saber donde se encuentra cada persona mediante un identificador. Debido a que

los identificadores son únicos, se puede pensar que el modelo de ubicaciones utilizado es similar a un modelo simbólico, donde las ubicaciones no poseen ningún tipo de relación entre si.

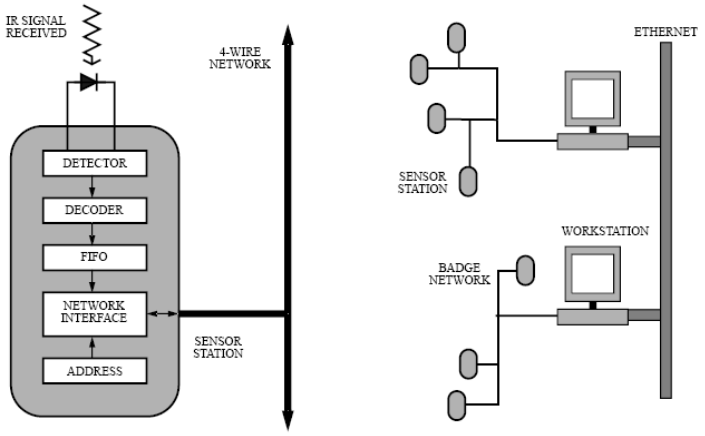


Figura 3.2: Sensores del Active Bat en una red de computadoras

Dentro de los sistemás de ubicación existentes, el Active Badge es conocido por ser uno de los primeros sistemas que dieron soporte para el manejo de ubicaciones. El sistema resuelve varios problemas que para la época eran importantes como son el consumo de batería, la emisión de señales y el rehúso de infraestructura computacional para permitir su integración con los sistemas de ubicación. Este último hecho no se debe de menospreciar ya que en esa época se utilizaban beepers con el fin de ubicar a una persona. Dichos dispositivos representaban un mundo aparte de las PC utilizadas en esos días en los centros de oficinas.

3.2.2. The Cricket Location Support System

El sistema Cricket [34] permite ubicar objetos dentro de ambientes cerrados al igual que el Active Badge [39]. A diferencia de este, uno de los objetivos de este sistema es que los objetos inferieran sus ubicaciones en base a las señales recibidas. Esto permite un mayor nivel de seguridad ya que cada objeto es el único que sabe su ubicación. En este caso, las ubicaciones son expresadas en términos de zonas. Las zonas son lugares donde las señales de un beacon (pequeños dispositivos que emiten identificadores únicos) son sensadas. De esta manera cada objeto podrá estimar las distancias a los beacons que esta escuchando y de esta manera estimar cual de ellos se encuentra más cercano.

Para permitir que cada objeto pueda inferir su ubicación, se utilizan un conjunto de beacons que se encuentran esparcidos geográficamente por el ambiente. Los beacons emiten dos tipos de señales, una señal de radio frecuencia y una

señal ultrasónica. En base a la diferencia en los tiempos de recepción de cada una de las señales es posible establecer la distancia con respecto al beacon emisor. Para lograr sensar dichas señales los objetos utilizan *listeners*. Cada listener, tiene la capacidad de escuchar múltiples beacons lo cual le permite establecer distancias a múltiples lugares de la organización. Las señales emitidas por los beacons se envían en forma concurrente y periódica de forma tal que un objeto pueda percibir al menos una señal en un instante de tiempo. Dado que no existe un mecanismo de sincronización a la hora de transmitir, las señales enviadas por los beacons pueden colisionar. En caso que este hecho ocurra, cada beacon utiliza una técnica de demora por un tiempo aleatorio para reenviar su señal al medio. Al igual que con el Active Badge, Cricket no detalla mucho acerca de como se han modelado las ubicaciones. En base a que la señal transmitida por cada beacon es un String, podría inferirse que se utiliza un modelo simbólico. En base a la ubicación de cada beacon y nuestra distancia con respecto a este, podremos determinar en que zona nos encontramos. Dentro de los sistemas desarrollados con Cricket, algunas aplicaciones han planteado extensiones al modelo original. FloorPlan por ejemplo, extiende el modelo agregando una correspondencia entre las ubicaciones simbólicas recibidas por los beacons y un conjunto de ubicaciones geométricas.

El sistema Cricket provocó un avance respecto a la posibilidad de ubicar objetos dentro de ambientes cerrados. Mejoró muchas de las características que poseía el Active Badge, permitiendo un alto nivel de seguridad. Esto lo logró utilizando la idea de que cada objeto es capaz de determinar su ubicación en base a un conjunto de señales. De esta forma se logró que otros objetos no conocieran la ubicación de uno en particular a menos que este la expusiera o trasmitiera.

3.2.3. The Lighthouse location system

El Lighthouse location system [35] hace uso del concepto de la smart dust [22] o basura inteligente. Esta basura se encuentra compuesta por un conjunto de microcomponentes que poseen un tamaño milimétrico con una limitada capacidad de cómputo que le permite determinar autónomamente su posición respecto a un conjunto de bases estacionarias. Su ubicación es calculada en base a un conjunto de distancias y ángulos que son obtenidos en base a las señales recibidas provenientes de las bases. El objetivo del sistema es permitir ubicar a estos pequeños objetos los cuales se pueden encontrar en cualquier parte. Su tamaño y bajo costo, permite colocarlos sobre cualquier objeto de interés.

El sistema no plantea un desafío de como las ubicaciones son modeladas pero si un hecho tecnológico del tamaño de los dispositivos. Debido a que las ubicaciones son computadas en base a ángulos y distancias, se intuye que el modelo de ubicaciones subyacente es geométrico. El trabajo se encuentra más abocado a una cuestión tecnológica respecto al tamaño de la basura de forma

tal que pasen desapercibidos y permitan obtener la ubicación de los objetos que la contienen.

3.3. Arquitecturas y Frameworks

Las arquitecturas y frameworks son el producto de haber logrado abstraer un conjunto de conceptos de varias aplicaciones. Las arquitecturas esquematizan una visión de alto nivel, la cual no se encuentra limitada a un caso en particular. Dicha arquitectura es luego instanciada en un framework que permite validar el esquema teórico presentado por la arquitectura. Es por ello que se realizará un resumen del conjunto de frameworks y arquitecturas que han sido utilizadas hasta el momento para el desarrollo de aplicaciones con ubicaciones.

3.3.1. Un modelo semi simbólico

Ulf Leonarhd propuso en su tesis de doctorado [25] una arquitectura para proveer servicios de ubicación a distintas aplicaciones. Dicha arquitectura propone una separación entre los aspectos de adquisición de ubicaciones, el modelo de ubicación y los servicios. Por este motivo dicha arquitectura se encuentra dividida en 3 capas:

- Sensado: La capa de sensado se encuentra subdividida en 3 capas. Por un lado se encuentra la capa de abstracción de sensores que son simples abstracciones de bajo nivel que interactuan con el hardware subyacente. La capa de recepción actuá como bus con el fin de independizar el conocimiento directo al sensor. La capa de abstracción unifica los datos de bajo nivel a un modelo único de datos, es decir extrae las particularidades de cada conjunto de datos dependientes del sensor en particular. Por último la capa de fusión, mezcla de varias fuentes información con el fin de proveer la ubicación combinada.

- Ubicaciones: La arquitectura utiliza algún modelo geométrico, simbólico o semi simbólico. Leonarhd introduce el modelo semi simbólico como una dupla de información simbólica y geométrica. De esta manera se provee un modelo simbólico que puede ser fácilmente comprendido por los usuarios y un modelo geométrico que debe mantener cierta coherencia respecto a los resultados de las operaciones con el modelo simbólico. El modelo es especificado mediante una dupla (ubicación simbólica, ubicación geométrica) y se proveen funciones de mapping entre cada representación geométrica con su correspondiente representación simbólica.

- Servicios: La capa de servicios esta subdividida en varias capas. Estas permiten que los consumidores de los servicios puedan llevar un control

de donde estuvo ubicado un objeto a traves del tiempo (*tracking*). También se permite interactuar con la capa de sensores con el fin de obtener información de posicionamiento. Por último, se proveen servicios que permiten consultar las relaciones espaciales que existen entre cada una de las ubicaciones.

Leonarhd agrega al trabajo factores de desconocimiento de ubicación, que son útiles cuando los sensores no proveen suficiente información de la ubicación de los objetos. También se agregan elementos de predicción de movimientos de los usuarios con el fin de determinar cuales serán los próximos lugares que el usuario estará visitando y de esta manera preveer cuales servicios se ofrecerán.

Lo novedoso del trabajo realizado por Leonarhd es la introducción del modelo semi simbólico para el modelado de las ubicaciones. Dicho modelo permite utilizar las ventajas de cada uno de los modelos subyacentes y opacar las desventajas de cada uno de ellos.

3.3.2. Alipes

Alipes [29] es una arquitectura orientada a resolver problemáticas del desarrollo de software que haga uso de ubicaciones. La arquitectura se encuentra dividida en un conjunto de capas (Figura 3.3) las cuales permiten atacar cada uno de los problemas encontrados en este tipos de aplicaciones por separado. Las capas presentadas por la arquitectura son:

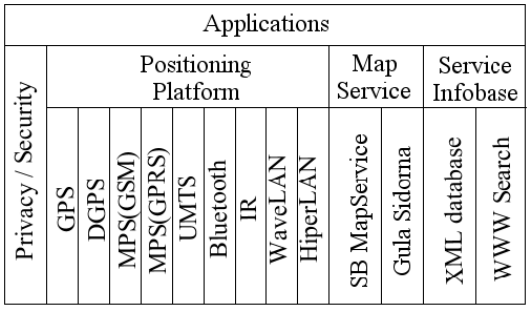


Figura 3.3: Arquitectura Alipes

- Privacy / Security: La capa de seguridad, permite definir contratos entra la aplicación que conoce una ubicación y el resto de las aplicaciones. De esta manera una aplicación corriendo sobre un dispositivo móvil como una PDA puede permitirle conocer su ubicación a otra aplicación cuando exista un contrato que la autorice. En este contexto, el contrato sirve a modo de un conjunto de reglas que se tienen que satisfacer para que la aplicación que esta solicitando la ubicación pueda conocerla.

- Map service: Utilizando el protocolo de posicionamiento genérico, las aplicaciones pueden solicitar al servicio de mapas, planos con diferente resolución. Esta capa tiene la responsabilidad de brindar los mapas necesarios a las aplicaciones que las soliciten tomando en cuenta que el detalle del mapa esta relacionado con las ubicaciones que desea mostrar la aplicación. Es decir, si la aplicación desea mostrar dos ubicaciones que se encuentran a km de distancia, el nivel de detalle del mapa deberá ser más bajo que si se encontraran a pocos metros ya que es probable que existan muchos objetos y su visualización sea complicada.

- Service Infobase: El servicio de información, provee información respecto de servicios publicados en internet que tengan que ver con aspectos relacionados con ubicaciones. Para ello las aplicaciones pueden solicitarle información respecto a todos los restaurantes en una zona y este servicio deberá proveer dicha información.

- Positioning Platform: Se encarga de obtener la información de cada uno de los dispositivos y combinar los valores obtenidos (mediante una política de merging) con el fin de obtener una ubicación más precisa. Por ejemplo en la figura 3.4 podemos observar como son mezcladas una figura geométrica obtenida por un GPS (un rectángulo) y otra obtenida por MPS[1] la cual es acotada entre dos círculos. Esta capa también tiene configurado como obtener la información de cada uno de los dispositivos mediante una política push o pull. La información obtenida es transformada en un nivel de abstracción más alto denominado protocolo de posicionamiento genérico. La idea de este protocolo es que sea legible por los seres humanos y esta basado en posiciones geométricas.

- Applications: Por último, las aplicaciones se ubican sobre estas capas haciendo uso de los servicios expuestos por las capas anteriores. Con el desarrollo de la arquitectura se construyeron dos aplicaciones. El FriendFinder es una aplicación simple que permite visualizar la ubicación actual del usuario en un mapa y visualizar la ubicación de sus amigos. Esta aplicación hace uso del servicio de mapas y de los contratos entre las aplicaciones para poder visualizar las ubicaciones. GeoNotes es una aplicación que permite dejar notas virtuales en lugares físicos. Para ello se publica información en el Service Infobase respecto de que nota ocupa que lugar físico.

La arquitectura desacopla correctamente el aspecto de sensado de como se modelan las ubicaciones. En el desarrollo de la misma no se detalla como se

[1] un mecanismo de sensado

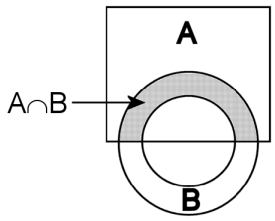


Figura 3.4: Merge entre la posicion dada por un GPS y un MPS

produce el proceso de transformación entre la información recibida y las ubicaciones. Las capas de servicios proveen servicios útiles a las aplicaciones, pero tampoco se explica como se acomodaría la arquitectura para proveer otro tipo de servicios más allá de los enunciados. La arquitectura no plantea un modelo de ubicaciones ni tampoco se detalla como haríamos para utilizarla en una aplicación existente. Igualmente, se puede deducir que el modelo de ubicaciones se encuentra altamente acoplado a los modelos geométricos, ya que la información sensada es transformada al protocolo de **posicionamiento** genérico. Como hemos visto en el Capítulo 2 los modelos geométricos no alcanzan para expresar todas las relaciones existentes entre objetos físicos por lo tanto la arquitectura no alcanza para modelar otras relaciones físicas entre los objetos.

3.3.3. The Location Stack

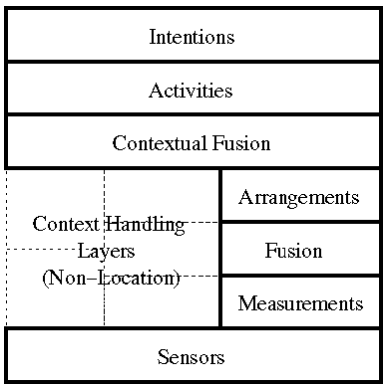


Figura 3.5: The Location Stack

El Location Stack [20] es una arquitectura que se encuentra basada en la misma idea que posee el modelo OSI [43] para redes. La arquitectura se encuentra dividida en siete capas (Figura 3.5) en forma de pila:

- Capa de sensado: La capa de sensado es la encargada de interactuar a bajo

nivel con el hardware. Provee de abstracciones de las señales recibidas en bytes de información.

- Capa de medidas: La capa de medidas es la encargada de transformar la información recibida por los sensores en información útil como ángulos y distancias. Es la encargada de transformar el conjunto de bytes en abstracciones de mayor nivel.

- Capa de fusión: La capa de fusión es la encargada de fusionar conjuntos de medidas eliminando incoherencias y redundancias.

- Capa de acomodamientos: La capa de acomodamientos es la que provee un mecanismo para razonar sobre las relaciones entre los objetos. También provee mecanismos para trasformar desde sistemas de ubicaciones geométricos locales a absolutos.

- Capa de fusión contextual: Esta capa se encarga de mezclar información de ubicación con otros aspectos de contexto como información temporal, temperatura, calor y humedad.

- Capa de actividades: La capa de actividades se encarga de inferir las actividades que están ocurriendo en base a la información contextual recibida. Para ello se pueden utilizar mecanismos de aprendizaje automático como redes neuronales o árboles de decisión.

- Capa de intensiones: La capa de intensiones muestra los deseos de los usuarios respecto de como se relacionan con el sistema o la tarea que esta ocurriendo.

Lo interesante de esta arquitectura es como se han extrapolado ideas de otras áreas para el desarrollo de una arquitectura para ubicaciones. El planteo utilizado es bueno ya que las capas de sensado, medidas y fusión pueden estar presentes o ser rehusadas de otros aplicaciones. La arquitectura fue utilizada para la implementación de dos aplicaciones que son casos de prueba para la arquitectura. Una de ellas es el Labscape que permite asistir en los experimentos a los científicos de un laboratorio de biología. Para ello la aplicación puede ir colaborando con el científico deduciendo que experiencia esta por realizar, preparando ciertas aplicaciones, preparados, etc. Otro aplicación desarrollada fue el de EasyLiving el cual permite un pseudo hogar inteligente proveyendo de encendido y apagado automático de luces y reproducción de música de acuerdo al lugar donde se encuentran los ocupantes del hogar. La arquitectura aún no provee un mecanismo para definir la representación del desconocimiento de la ubicación. A pesar de que la división en capas resulta razonable, la misma presenta ciertos aspectos que todavía no se dejan en claro en el trabajo. Por ejemplo, como podemos

agregar semánticas a las ubicaciones y como representamos el desconocimiento de la ubicación.

3.3.4. Realms and States

Realms and States [26] propone la idea de trabajar lógicamente con las ubicaciones. Para ello se proponen dos conceptos, un Realm representa una forma de particionar el espacio, como por ejemplo todos los estadios de deportes. Un State representa un elemento dentro de ese conjunto, por ejemplo dentro del Real todos los estadios de deportes, La Bombonera representa un State. La arquitectura presenta un conjunto de funciones llamadas Realm-maps que permiten mapear states desde un realm a otro realm. Esto permite que podamos particionar el espacio (Realm) y que podamos operar con ubicaciones (State) de distintos Realms utilizando funciones (Realm-maps) para unificarlas a un Realm en común. A bajo nivel, la arquitectura utiliza raw locations, las cuales son ubicaciones que han sido obtenidas desde bajo nivel por los sensores. Las raw locations son generalizadas a nivel de state mediante un conjunto de funciones de transformación. Tanto los Realm-maps como las operaciones poseen una definición formal en la arquitectura.

El aspecto interesante de esta arquitectura es que sigue el proceso cognitivo que los seres humanos realizamos al particionar el espacio y razonar sobre espacios más pequeños. Los mappings entre los diferentes espacios permiten trabajar indistintamente con cualquiera de los States de los objetos. Sin embargo, el mecanismo de mappings es un mecanismo que no escala cuando necesitamos trabajar con n objetos que poseen el mismo significado, ya que necesitamos n^2 mappings para asegurar la igualdad semántica de cada uno de ellos.

3.3.5. Location$_+$

La ubicación de los objetos es utilizada con el fin de mostrar un mapa con figuras geométricas. El enfoque adoptado en el modelo Location$_+$ [14] intenta mejorar dicha idea enriqueciendo una ubicación con información referente a otros aspectos del dominio. Por ejemplo, una ubicación podría ser enriquecida con información sobre el código postal de la ciudad que la contiene o una descripción del lugar donde se encuentra ubicado. Esto plantea una alternativa a la solución que se utiliza generalmente en este tipo de problemas, la cual utiliza mappings que relacionan ubicaciones con información.

Desde el punto de vista conceptual el modelo Location$_+$ introduce la idea de que una ubicación no sólo como una posición en un mapa sino como una ubicación con información. Sin embargo, el modelo no describe los objetos ubicables lo cual fuerza a agregar información a la ubicación. Esto es totalmente objetable ya que podrían existir dos objetos con la misma ubicación pero con

diferente información. Por ejemplo, la ubicación de dos personas podría ser la misma, pero la información que podría tener la aplicación respecto de cada una podría ser diferente.

3.3.6. Ubicaciones semánticas

Las ubicaciones semánticas [9, 33] intentan dar semántica a ubicaciones independizándolas de como están representadas. Para el modelado de las mismas existen dos formas:

- Una de las formas propuestas para implementar las ubicaciones semánticas es mediante una URI. De esta forma los modelos de ubicaciones conocidos anteriormente son mapeados a dicha ubicación con el fin de mantener la univocacidad del concepto.

- Otra opción que a primera vista resulta más complicada es la de establecer un conjunto de grafos donde las aristas definen la semántica de las operaciones. La ventaja de este modelo es que puede ser creado y mantenido por una computadora evitando la construcción manual que presenta un modelo simbólico. Para este modelo se definen dos grafos: uno que modela las ubicaciones con las relaciones que determinan sus operaciones y por otro lado, un grafo de salidas que especifica las salidas entre dos ubicaciones. Una salida es una ubicación que es compartida entre dos ubicaciones. Este último modelo presenta un conjunto de definiciones formales, basadas en teoría de grafos, que permite establecer un conjunto de propiedades que debe satisfacer el modelo. Esta formalización permite garantizar que las operaciones entre diferentes modelos sean aplicadas correctamente.

La idea que introduce el concepto de ubicación semántica es interesante desde un punto de vista conceptual, ya que define una ubicación como algo más general que una figura o un símbolo. Sin embargo al intentar modelar el concepto se utilizan URIs como mecanismo de mantener la unicidad entra las ubicaciones. Esto representa un punto donde el modelo es cuestionable, ya que se puede indagar que su uso es utilizado con fines de desarrollar una aplicación web. Existen otros mecanismos para mantener unicidad como son los números y los strings, por lo cual el modelo debería dar un fundamento más solido y convincente respecto de su uso. Por otro lado, no se explica con que objetivo se utiliza una URI como mecanismo de modelar una ubicación, ni como se responde a las operaciones si una URI posee más de una ubicación geométrica o simbólica asociada.

3.3.7. NEXUS

NEXUS [27, 28] es una arquitectura abierta que tiene como objetivo brindar información para el desarrollo de aplicaciones que hagan uso de la ubicación. Para ello la arquitectura propone una división en 3 capas (Figura 3.6):

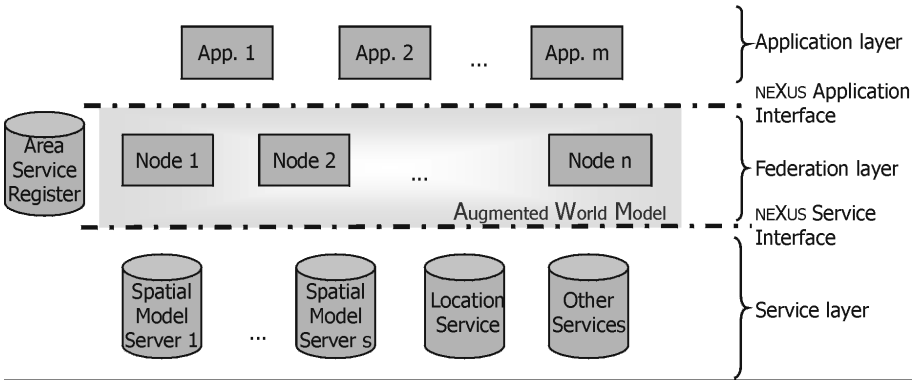


Figura 3.6: Arquitectura NEXUS

- Aplicación: La capa de aplicación representa las aplicaciones de un dominio en particular que están interesadas en servicios basados en ubicaciones.
- Federación: La capa de federación es utilizada para pegar la capa de aplicación con la capa de servicios. Es la encargada de distribuir la carga ante un nuevo requerimiento y de volver a unir el resultado para que este sea devuelto correctamente a la capa de aplicación.
- Servicios: La capa de servicios es la encargada de brindar los servicios de acuerdo a las ubicaciones solicitadas.

El enfoque adoptado particiona el conjunto de ubicaciones en augmented spaces de manera tal que la información pueda ser fácilmente distribuida en cada BD. Cada espacio aumentado (augmented space) define un conjunto de ubicaciones que poseen un sentido lógico de estar juntas. Por ejemplo, un espacio aumentado podrían ser todas las ubicaciones de la Facultad de Informática de la UNLP. Otro espacio aumentado serían todas las ubicaciones del museo Dardo Rocha. Para poder recuperar este conjunto de objetos se definió un nuevo lenguaje AWQL, el cual esta basado en XML. De esta manera se permiten recuperar los objetos pertenecientes a los espacios aumentados con el fin de obtener información de sus ubicaciones.

Esta arquitectura se encuentra instanciada en un Framework (Figura 3.7) que provee de un conjunto de clases base que pueden ser utilizadas para instanciar una aplicación NEXUS. Dentro de las clases desarrolladas se pueden

distinguir entre aquellos objetos que se encuentran ubicados en el espacio y son estáticos (no se pueden mover) de los móviles. En el caso que necesitemos extender el framework deberemos subclasificar de StaticObject en el primer caso y de MobileObject en el segundo.

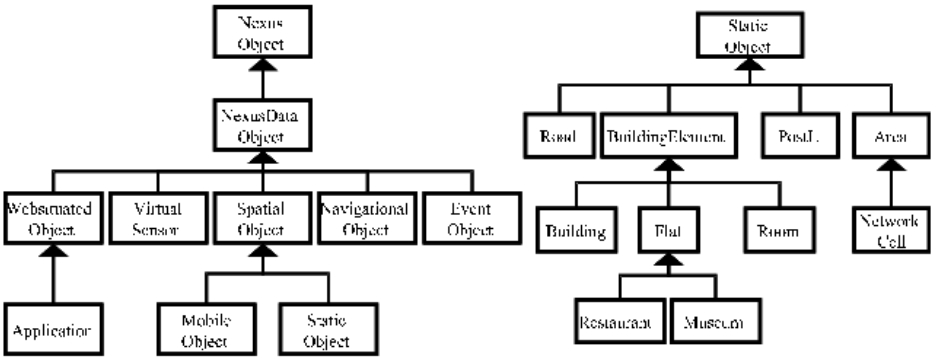


Figura 3.7: Diagrama de clases base de NEXUS

El enfoque adoptado por NEXUS presenta un problema de separación de concerns respecto del concern de persistencia con el de dominio de la aplicación. La arquitectura se encuentra totalmente acoplada a una federación de BDs relacionales. Por otro lado, las clases base presentadas no se encuentran correctamente modeladas ya que el concepto de herencia esta incorrectamente utilizado. Por ejemplo una aplicación es subclase de WebSituatedObject, lo cual es incorrecto ya que no permite modelar aplicaciones existentes. El concepto debería haberse modelado como una relación de conocimiento que le permita ser accedida desde la web.

Capítulo 4

Abstracción de la ubicación

4.1. Concepto de ubicación

Los modelos geométricos y simbólicos (Capítulo 2) han sido utilizados para modelar ubicaciones en numerosas aplicaciones. Sin embargo, debido a los requerimientos cambiantes de las aplicaciones, estos modelos han quedado obsoletos y han provocado el desarrollo de nuevos modelos que permiten hacer frente a estos cambios (Capítulo 3). Entre ellos podemos encontrar a los modelos semánticos que modelan a la ubicación como una URL y a los modelos semi simbólicos que permiten combinar ubicaciones geométricas y simbólicas en forma de dupla.

Dichos modelos tienen como objeto de análisis a la ubicación. Cada uno de ellos permite modelar este concepto de diferentes formas. Los modelos geométricos permiten modelar una ubicación como un punto, un polígono o una polilinea mientras que los modelos simbólicos permiten modelar a las ubicaciones como símbolos con relaciones semánticas entre estos. Con el objetivo de obtener los beneficios de ambos modelos, Leonarhd desarrollo los modelos semi simbólicos que tenían como objetivo combinar en forma de dupla una ubicación simbólica y una ubicación geométrica. Por último las ubicaciones semánticas fueron introducidas con el objeto de definir ubicaciones utilizando URLs y mappings. En este modelo la relación de inclusión esta definida por comprensión aunque el resto de las operaciones pueden ser definidas mediante relaciones. El uso de mappings permite mantener la equivalencia semántica con otra ubicaciones geométricas y simbólicas provistas por otros modelos.

Estos modelos intentan expresar un subconjunto de relaciones físicas existentes entre las ubicaciones. Es decir, intentan **representar** a la ubicación en cada uno de ellos con el objetivo de que sea el modelo quien permita representar algunas de las relaciones físicas de interés. Por ejemplo, la ubicación de una persona en una ciudad puede ser representada en un modelo geométrico como un punto

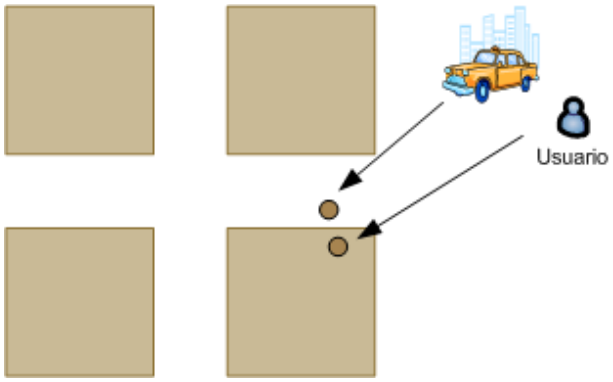


Figura 4.1: Una persona y un auto en un modelo geométrico de la ciudad

(Figura 4.1). Sin embargo, la persona puede tener otras relaciones físicas que no surjan naturalmente de dicho modelo. En este caso, la ubicación de la persona modelada como un punto no nos dice si la misma es o no adyacente al auto. Podriamos querer expresar dicha relación aunque el modelo no sea conveniente. En tal caso utilizaríamos un modelo simbólico que nos permite representar a la ubicación de la persona y del auto permitiendo definir la relación de adyacencia entre ambos.

El ejemplo de la persona y el auto nos permite visualizar que el modelo no es más que una forma de representar las ubicaciones. Si desacoplamos el concepto de ubicación respecto de su representación podremos trabajar con la ubicaciones independientemente de como estas se encuentren representadas en uno o varios modelos. Las relaciones físicas entre las distintas ubicaciones quedan determinadas por las relaciones que existen entre las formas que hemos optado para representarlas en los distintos modelos. Es decir, volviendo al ejemplo, si la ubicación del auto y de la persona fueron modeladas simbólicamente, entonces la afirmación que dice que el auto es adyacente a la persona sería correcta, ya que una de las formas que hemos representado a la ubicación de dichos objetos hemos expresado dicha relación.

4.2. Representación

En los modelos anteriores la ubicación fue modelada de diferentes formas. Por este motivo hablaremos de la representación de una ubicación en un modelo o simplemente **representación** a la forma en la cual una ubicación es representada en un modelo.

Volvamos al ejemplo de la persona en la ciudad junto al auto. Esta situación podría haber sido modelada de distintas formas dependiendo de la persona que construye el modelo. Por ejemplo, tenemos un modelo geométrico de alto nivel

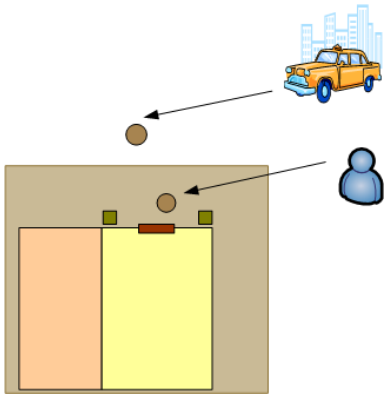


Figura 4.2: Zona de entrada al edificio

en donde se modelan las manzanas, las calles, el auto y la persona. En forma similar, otra persona desarrollo otro modelo que permite definir relaciones de adyacencia que no son posibles de determinar en el modelo geométrico. Por otro lado, el edificio cercano a la persona ha introducido un modelo que permite dar un nivel de detalle mayor a la zona de la entrada del edificio (Figura 4.2). Si continuamos con este proceso podremos observar que distintas personas podrían modelar distintos modelos que tengan el mismo significado. Este hecho nos permite deducir que cada modelo nos da diferentes visiones de las relaciones físicas entre las ubicaciones. Cada modelo puede aportar relaciones que ya son conocidas o que enriquezcan las ubicaciones existentes. Es decir, el mecanismo que nos permite aumentar el conocimiento respecto a las relaciones físicas de una ubicación, son las representaciones que una ubicación posea en los distintos modelos.

Figura 4.3: Edificio simbólico

Supongamos que una empresa posee oficinas dentro del edificio (polígono amarillo) y que ha confeccionado un modelo simbólico (Figura 4.3) del interior

del mismo. La puerta principal del edificio se encuentra representada en ambos modelos de diferentes formas. Cada una de ellas permite deducir diferentes relaciones físicas entre las ubicaciones. Con el conocimiento del nuevo modelo, ahora sabemos que no sólo la puerta se encuentra a 5 metros de la calle sino que también la puerta es adyacente al hall del edificio (asumiendo que el simbolo #hall representa la ubicación del hall del edificio). Por lo tanto, agregar semántica a las ubicaciones es tan sencillo como agregar nuevos modelos que permitan deducir nuevas relaciones físicas entre ellas.

Muchas veces puede surguir la duda respecto de como operar entre distintas representaciones. Por ejemplo, como podemos calcular la distancia entre una representación simbólica y un punto en un modelo geométrico. En principio podemos pensar que esto no tiene mucho sentido ya que son distintos tipos de representaciones. Sin embargo, es cierto que podemos operar entre dos representaciones geométricas que no necesariamente se encuentren en un mismo sistema de representación. Por tal motivo diremos que las representaciones operan entre si siempre y cuando sean compatibles. Es decir, no tiene sentido intentar operar entre una representación simbólica y una representación geométrica ya que las operaciones están definidas de diferente forma en cada modelo. Por lo cual, las operaciones entre las representaciones sólo se podrán realizar si las representaciones pueden ser expresadas en un modelo en común que les permita operar.

4.3. Sistema de representación

Los modelos geométricos y simbólicos nos permiten modelar ubicaciones geométricas y simbólicas respectivamente. Por este motivo utilizaremos el termino sistema de representación (SR) con el objeto de referirnos a los modelos que nos permiten representar a las ubicaciones. De esta forma tendremos sistemas de representación simbólicos y geométricos, los cuales nos permitirán generar representaciones simbólicas y geométricas.

Volviendo al ejemplo, supongamos que nos referimos al simbolo #puerta. La pregunta que surgue es ¿ Donde se encuentra definido #puerta ? o ¿ Qué pasa si #puerta esta definido en varios sistemas de representación simbólicos ?. Para ello utilizaremos el mismo esquema que se utiliza con los Namespaces [6]. Los sistemas de representación definen el ámbito donde las representaciones se encuentran definidas. En el ejemplo, el símbolo #puerta se encuentra definido en el sistema de representación simbólico provisto por la empresa pero no en el modelo simbólico del auto y la persona. Debido a que el sistema de representación nos define el ámbito donde se encuentra definida una representación, este deberá permitirnos crear las representaciones en ese ámbito. Por ejemplo, un sistema de representación geométrico nos deberá permitir construir puntos,

polilineas y polígonos dentro de ese ámbito.

Entre los sistemas de representación geométricos podemos definir relaciones que permitan transformar un elemento de un sistema a otro (Capítulo 2). Por este motivo cuando nos referíamos a que dos representaciones podrán operar entre si, es decir que sean compatibles, el sistema de representación geométrico podrá transformar una representación de su sistema a una representación que se encuentre en otro utilizando dicha transformación. Como ejemplo, supongamos que entre el sistema de representación geométrico de la zona del edificio y el sistema geométrico de la ciudad hemos definido una relación de traslación entre ambos. En este caso si deseamos operar entre una representación definida en el sistema de representación de la ciudad y una representación definida en el en el sistema de representación de la zona del edificio, deberemos transformar las representaciones a un sistema de representación común con el fin de que podamos operar.

4.4. Reacomodando conceptos

Debido a que se han introducido dos conceptos nuevos, deberemos acomodar los conceptos introducidos anteriormente (Capítulo 2) con el fin de que los mismos sean coherentes. Por este motivo se detallará como estos se acomodan a los enunciados anteriormente.

4.4.1. Ubicación

El concepto de ubicación fue descripto en base a sus objetivos y/o propiedades (Capítulo 2). Esta forma de describir a una ubicación nos permitió independizarnos de como ésta se encontraba modelada. Por lo tanto, el concepto de ubicación como una abstracción no ha cambiado con la introducción de las representaciones. Sin embargo, la forma en la cual nos referimos a los modelos es lo que **si** ha cambiado. Anteriormente, las ubicaciones eran modeladas **sólo** como una figura geométrica, un símbolo, una dupla o una URL. Esto fuerza a que el modelo se defina en forma rígida al igual que las definiciones alternativas que proponían Wikipedia y la Real Academia Española. Luego, con la idea de expresar que es una ubicación en base a las cosas que esta puede satisfacer, relajamos el concepto. La introducción del concepto de representación permite que una ubicación puede ser representada de múltiples formas en diferentes sistemas de representación aumentando, de esta forma, el conjunto de relaciones físicas conocidas. Por ejemplo, la puerta del edificio se encuentra representada de dos formas diferentes, mediante un polígono en el SR geométrico y un símbolo en el SR simbólico permitiendo que cada una de ellas aporte el conocimiento de alguna nueva relación. En base a este analisis podemos ver que la representación es lo que nos permite, a nivel de modelo, relajar el concepto de ubicación evitando

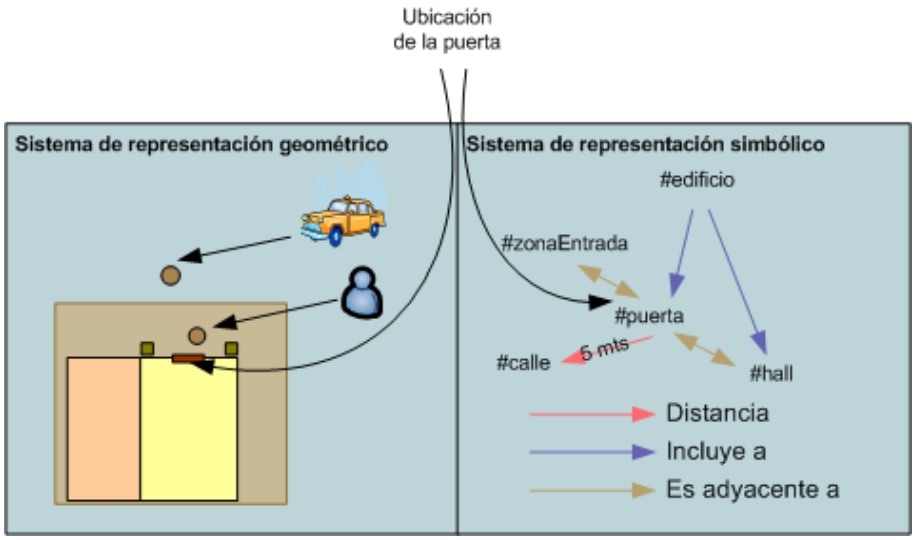


Figura 4.4: Puerta representada de múltiples formas

que este se encuentre rígidamente definido, permitiendolo variar de acuerdo a como este se encuentre representado.

Una pregunta que surge de la figura 4.4 es: ¿ Cómo computar las operaciones entre una ubicación que se encuentra múltiplemente representada, como es el caso de la puerta del edificio ?. Por dicho motivo tenemos que definir como se computará una operación si varias representaciones permiten dar semántica a la misma. Aquí las opciones son varias y pueden depender del tipo de aplicación que se este desarrollando. Para el caso de la operación de distancia podemos optar entre elegir la menor, el promedio o la mayor. Por otro lado las operaciones de inclusión y adyacencia que responden con verdadero o falso se puede exigir que al menos una representación responda si para que la respuesta sea válida. Notemos como este hecho se asemeja a lo que ocurre en la realidad. Por ejemplo, para determinar a que distancia física se encuentran dos objetos se realizan N medidas con diferentes aparejos que pueden resultan en diferentes resultados. La persona que realiza las medidas puede estimar la distancia utilizando el promedio, el mayor o el menor de los resultados.

4.4.2. Sistema de ubicaciones

Los sistemas de ubicación o sistemas que dan soporte para el trabajo de ubicaciones se ocupaban de abstraer las señales eléctricas, infrarrojas o sonoras con el fin de modelar a las ubicaciones. Debido a que hemos cambiado la forma en la cual las ubicaciones se modelan, deberíamos decir que los sistemas de ubicaciones dan soporte para la generación de representaciones en los diferentes

sistemas de representación. Esto es debido a que las señales se abstraen al nivel de símbolo o figura geométrica que en nuestro contexto no son ubicaciones sino que son formas de representar una ubicación en un sistema de representación particular. Por dicho motivo los sistemas de ubicación serán los responsables de crear las representaciones en base a las señales recibidas.

4.4.3. Objeto ubicable

El término objeto ubicable fue introducido por Schilit [36] con el objetivo de referirse a los objetos que poseían una ubicación. Dado que la definición no hace referencia a que modelos se utilizan ya sea geométricos o simbólicos, la definición es válida también en nuestro contexto.

4.4.4. Mapa

Un mapa no muestra un conjunto de objetos ubicables en un plano geométrico (Capítulo 2). Es decir, el mapa tiene como objetivo mostrar un conjunto de objetos ubicables en un sistema de representación en particular permitiendo que la persona que lo vea pueda interpretar las relaciones físicas entre los objetos. Desde el punto de vista de las ubicaciones y sus representaciones podemos ver que un mapa como lo conocemos, nos permite mostrar las representaciones que tienen las ubicaciones de los objetos en un sistema de representación geométrico.

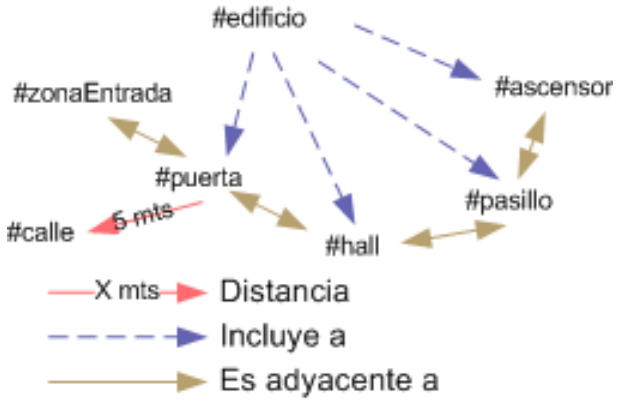


Figura 4.5: Mapa simbólico

El hecho de que un mapa tenga como objetivo mostrar un conjunto de objetos con sus relaciones físicas, no debe forzar la idea de que este deba ser representado en un plano geométrico. Podríamos crear un mapa con un conjunto de símbolos y flechas que nos digan las relaciones físicas que existen entre los objetos sin que ello involucre aspectos geométricos (Figura 4.5). De este hecho surge la necesidad de referiremos a un **mapa abstracto**, como un conjunto de objetos

ubicables que tiene como objetivo mostrar las relaciones físicas existentes entre ellos. Hablamos en el sentido abstracto ya que no afirmamos si los objetos serán representados mediante figuras geométricas o símbolos u otras formas de representar ubicaciones. Es decir, los mapas abstractos ven a los objetos ubicables con sus ubicaciones abstractas sin importar como estas se encuentren representadas.

Los mapas que se visualizan sobre un plano geométrico son casos particulares de los mapas abstractos. Un mapa que se dibuja sobre un plano geométrico intenta mostrar a los objetos ubicables en un sistema de representación en particular. Por este motivo, este tipo de mapas serán llamados **mapas concretos** ya que permite visualizar a los objetos ubicables de un mapa abstracto en un sistema de representación en particular. Tendremos mapas concretos que permitan visualizar a los mapas abstractos tanto en un sistema de representación geométrico como en un sistema de representación simbólico.

4.4.5. Modelos básicos de ubicaciones

Los modelos básicos que vimos anteriormente (Sección 2.2) nos permitían modelar ubicaciones como figuras geométricas o símbolos. Dichos modelos lo que estaba haciendo en realidad era representar a la ubicación en el modelo. Es decir, las figuras geométricas y los símbolos son dos formas diferentes de representar una ubicación. El modelo simbólico y los modelos raster, spaghetti y topológico (Sección 2.2.1) son distintos sistemas de representación. El sistema de representación simbólico representa a la ubicaciones como símbolos. Los distintos modelos geométricos representan a las ubicaciones como figuras geométricas en el plano. Es decir los modelos son nuestros sistemas de representación.

El modelo semi simbólico introducido por Leonarhd (Sección 3.3.1) utiliza una dupla (ubicación simbólica, ubicación geométrica) para modelar a las ubicaciones. Este modelo representa un caso particular del esquema planteado para modelar a las ubicaciones. El modelo semi simbólico sería en nuestro caso un objeto con una ubicación que posee dos representaciones: una geométrica y otra simbólica. Por lo tanto, el modelo presentado por Leonarhd representa un caso particular del uso del concepto de representación, aunque en su contexto se utilizaba el término ubicación en lugar de representación.

Las ubicaciones semánticas modelaban a las ubicaciones como URIs (Sección 3.3.6). En base a la información que se podía capturar de la URI se podía establecer una relación de inclusión entre las ubicaciones. Las ubicaciones semánticas también permitían establecer mappings con las ubicaciones geométricas y simbólicas con el fin de mantener la equivalencia semántica entre las mismas. Este modelo utiliza URIs como forma de representar a la ubicación. Si observamos detenidamente, la URI es un string o símbolo que posee cierta semántica dada por comprensión. Por ello las ubicaciones semánticas son formas de representar simbólicamente a las ubicaciones, en donde las operaciones son dadas por

comprensión sobre los símbolos. Este modelo es desarrollado, en nuestro caso, como una ubicación que posee una representación simbólica en forma de URI y desde ninguna a varias representaciones más en forma geométrica o simbólica (Figura 4.6).

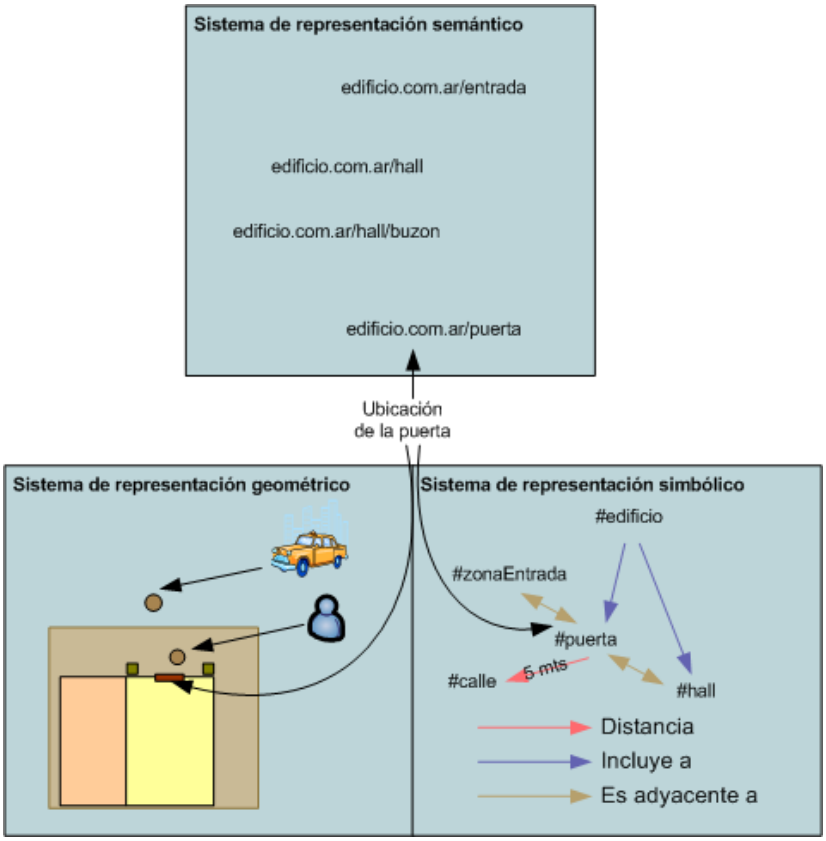


Figura 4.6: Puerta con representación semántica

Capítulo 5

Una arquitectura para sistemas de ubicación

5.1. Alcance del trabajo

En los capítulos anteriores se mostró como utilizando símbolos o figuras geométricas podíamos modelar ubicaciones. Sin embargo dichos modelos no permiten una fácil adaptación e integración con otros modelos. Luego realizamos un análisis conceptual de los términos (Capítulo 4) y en base a ello obtuvimos una idea alternativa de como modelar las ubicaciones. Esta idea se basa en que los modelos anteriores están conceptualizando diferentes representaciones de la misma ubicación. Dichas representaciones son las que dan la semántica a las operaciones de las ubicaciones. Si modelamos a la ubicación como un ente abstracto y permitimos dar semántica a sus operaciones por medio de como esta se encuentre representada entonces podremos dar solución a los problemas de escalabilidad e integración anteriormente enunciados.

En base a estas ideas se planteó una nueva arquitectura que pudiera hacer frente a los problemas enumerados en el capítulo 1. La misma dará soporte a sistemas de ubicaciones haciendo principal hincapié en los siguientes puntos:

- Debe dar el soporte necesario para el trabajo con ubicaciones permitiendo que la misma pueda escalar fácilmente. Es decir si deseamos enriquecer el modelo de ubicaciones con otras representaciones de la misma ubicación esto podrá ser realizado sin necesidad de tener que parar la aplicación.
- Debe permitir que modelos de ubicaciones desarrollados en forma separada sean fácilmente integrados. Por ejemplo, si dos personas desarrollan por separado el modelo de ubicaciones de la ciudad de La Plata y de la Facultad de Informática de la UNLP, este último podrá ser integrado con el primero de una forma sencilla.

- El soporte para ubicaciones se podrá agregar a cualquier aplicación existente. Esto permitirá que una aplicación que se encuentre en funcionamiento pueda beneficiarse del uso de ubicaciones. El área de aplicaciones no sólo toma en cuenta a las aplicaciones Context Aware sino que cualquier aplicación que necesite utilizar ubicaciones podrá beneficiarse del uso de la misma.

Por otro lado, este trabajo no tomará en cuenta los siguientes puntos:

- El sensado de la ubicación no será un aspecto de interés para la arquitectura ya que existen trabajos [15, 23] que resuelven este problema permitiendo modelar y transformar los valores sensados en objetos de mayor nivel. Sin embargo, se mostrará con que capas de la arquitectura deberá interactuar un framework de sensing.

- El producto final de la utilización de ubicaciones tampoco será un aspecto de interés. De esta forma no forzamos a la utilización de una capa de servicios forzando a que las aplicaciones adopten una arquitectura en base a servicios cuando no es necesario.

5.2. Soporte para el framework desarrollado

Como se planteo en la sección anterior, el framework desarrollado debe permitir la introducción de la ubicación sobre un sistema existente con el mínimo de intrusión posible. Esto motivo a que se haya adoptado un enfoque de objetos puros, en donde los conceptos de programación orientada a objetos sean manejados uniformemente. Los lenguajes híbridos como C++ y Java no manejan uniformemente los conceptos de mensaje y que todo elemento del ambiente es un objeto. Por estos motivos se optó por implementar al framework utilizando un ambiente como Smalltalk, en particular la distribución VisualWorks 7.4 [5].

5.3. Un escenario representativo

Las aplicaciones que pueden necesitar hacer uso de ubicaciones pueden pertenecer a distintas áreas y se pueden encontrar en diferentes etapas de desarrollo (en producción, en desarrollo, en testing, etc). A modo de ejemplo, supongamos que en la Facultad de Informática de la UNLP se desarrollará un congreso y que para dicha ocasión se desea desarrollar una aplicación que brinde información sobre las exposiciones de cada una de las publicaciones. Dicha aplicación se ejecutará sobre una PDA y brindará servicios basados en la ubicación del usuario. Para dicho acontecimiento la facultad ha desarrollado una aplicación que le permite administrar y organizar el lugar de cada una de las exposiciones, en que

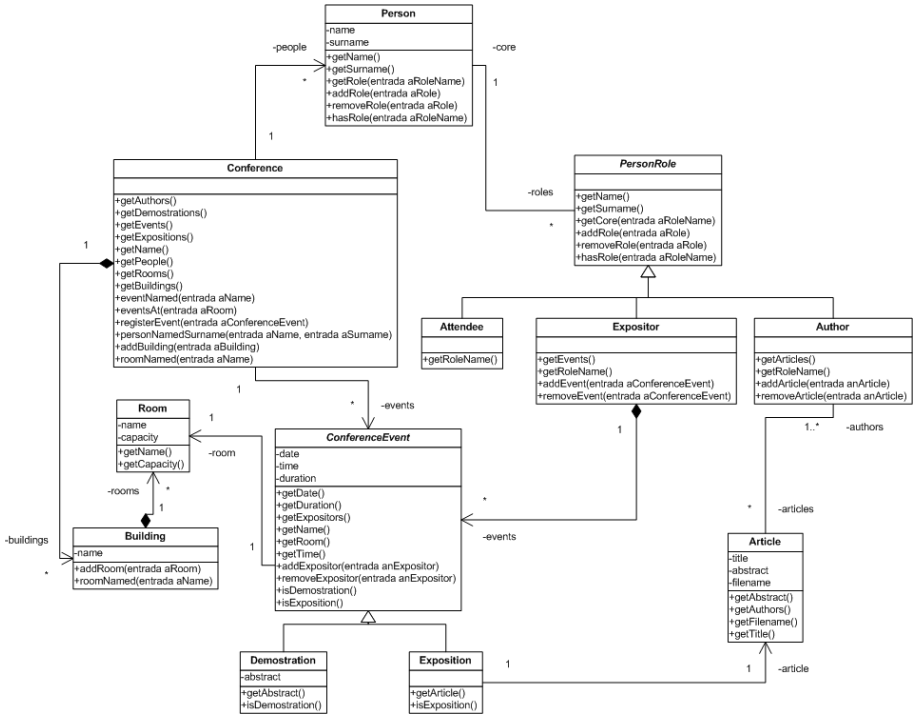


Figura 5.1: Modelo del sistema de la conferencia

horarios se llevarán a cabo y las personas que están registradas en el evento. Dicho modelo (Figura 5.1) posee modeladas a las personas que se van a presentar al congreso en carácter de oradores y las personas que se encuentran registradas en carácter de asistentes. También las aulas y los edificios que contienen a las aulas se encuentran modeladas, los eventos: demostraciones o publicaciones que se desarrollaran en cada una de las aulas. El uso de este sistema le permite a la Facultad hacer un uso eficiente de cada una de sus instalaciones.

En base a este modelo, desarrollaremos una aplicación que permitirá que las personas que se encuentran en el congreso puedan realizar un conjunto de actividades basadas en su ubicación. Por ejemplo, cuando un usuario ingresa a un aula podría visualizar en su dispositivo móvil el contenido de la publicación que esta siendo proyectada. Dicha persona podría haber organizado a que charlas concurriría y en base a ello podrá solicitarle a la aplicación que le indique como llegar hasta cada uno de los lugares donde se expondrán las publicaciones. Por otro lado, la aplicación debe brindar la asistencia necesaria a los usuarios que necesiten ser guiados hacia lugares de comidas o alojamiento. Por último, la Catedral de La Plata expondrá un mapa con el fin de que los usuarios que visiten dicho lugar puedan ubicarse dentro de la misma.

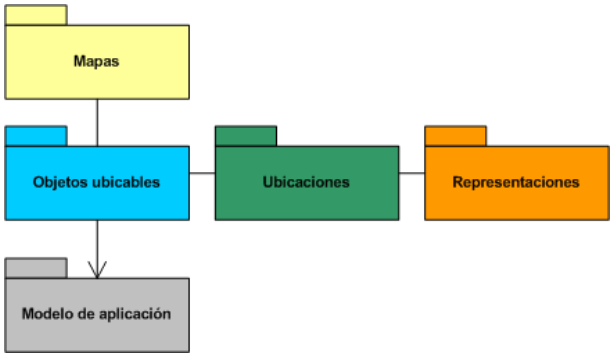


Figura 5.2: Arquitectura para sistemas de ubicación

5.4. Arquitectura

Los modelos de ubicaciones desarrollados en los capítulos 2 y 3 intentan modelar a la ubicación como un ente abstracto que puede ser formalizado y perfectamente delimitado. En el capítulo 4 planteamos una visión alternativa que intenta explicar que es una ubicación en base a las propiedades que puedo expresar por medio de sus representaciones. La arquitectura desarrollada (Figura 5.2) toma como base esta idea con el objetivo de dividir en capas cada uno de los aspectos básicos del trabajo con ubicaciones. La división en capas [4] permite dividir claramente cada uno de los aspectos de la arquitectura y a su vez delimitar las responsabilidades de cada una de las capas. A continuación describiremos cada una de las capas de la arquitectura y como estas interactúan.

5.4.1. Capa de aplicación

La capa de aplicación es la capa que permite modelar los objetos propios del dominio de la aplicación. En nuestro ejemplo, las personas, las aulas y las publicaciones están modeladas en esta capa. Los objetos pertenecientes a esta capa no se encuentran ligados a aspectos de las ubicaciones. Es decir sólo toman en cuenta el comportamiento propio del dominio, como puede ser registrar una persona, consultar una publicación, ver el cronograma, etc.

5.4.2. Capa de objetos ubicables

La capa de objetos ubicables permite enriquecer los objetos del modelo de dominio con ubicaciones. Recordemos que el término objeto ubicable fue introducido por Schilit [36] para referirse a aquellos objetos que poseen una ubicación. Es decir un objeto como un aula perteneciente a la capa de aplicación se enriquece con otro objeto perteneciente a la capa de objetos ubicables que le permite

tanto agregar la noción de ubicación como el comportamiento necesario para realizar operaciones sobre las ubicaciones.

En esta capa pueden existir otros objetos que no hayan sido modelados en la capa de aplicación. Por ejemplo los pasillos no eran de interés para la aplicación del congreso, pero si necesitamos guiar a las personas dentro de los edificios deberemos tener conocimiento de como ir de un aula a otra. Los pasillos y las zonas de los edificios deberán ser modelados en esta capa ya que necesitaremos enriquecer el modelo básico con objetos más representativos del modelo de ubicaciones del lugar.

5.4.3. Capa de ubicaciones

En base a las ideas introducidas en el capítulo 4, la capa de ubicación deberá permitir trabajar con las ubicaciones en forma abstracta. Los objetos ubicables harán uso de las ubicaciones independientemente de como estas se encuentren representadas. Luego, la capa de ubicación deberá interactuar con la capa de representación con el objetivo de definir la semántica de las operaciones.

En las aplicaciones con dispositivos móviles es común que se desconozca la ubicación del usuario cuando los sensores no nos permiten determinarla. Por ejemplo, si la PDA sólo tiene capacidad de sensar una señal de GPS y el usuario ingresa en un aula o un edificio no conoceremos su ubicación ya que el GPS no funciona en interiores. Por este motivo la capa de ubicación deberá proveer algún mecanismo para especificar el desconocimiento de la ubicación.

La capa de sensado hará uso de esta capa para instanciar nuevas ubicaciones en base a las señales recibidas. De esta forma se creará una nueva ubicación cada vez que los sensores determinen que la ubicación de una persona ha cambiado.

5.4.4. Capa de representaciones

La capa de representación permitirá representar a la ubicación en cada uno de los modelos básicos expuestos en el capítulo 2. Esta capa deberá proveer las representaciones para las ubicaciones en los modelos básicos existentes y en nuevos modelos que se desarrollen.

En nuestro ejemplo, tendremos diferentes sistemas de representación para la ciudad, los edificios donde se desarrollará el congreso e incluso para la Catedral de La Plata.

La capa de sensado hará uso de esta capa para crear las representaciones de las ubicaciones cuando la señal sensada cambie. Por ejemplo, cuando la señal de un GPS cambie, la capa de sensado deberá instanciar una nueva representación que represente a la ubicación como un punto en un sistema geométrico.

5.4.5. Capa de mapas

Por último, la capa de mapas permite tomar un conjunto de objetos ubicables relacionados con el objetivo de proveer una vista parcial sobre el universo de objetos ubicables. Esta visión sobre los objetos ubicables es independiente de cómo se encuentren representadas sus ubicaciones. Para tomar una visión particular de los mapas se utilizarán los mapas concretos que permiten ver un mapa (abstracto) desde un sistema de representación en particular.

La capa de mapas deberá permitir trabajar con los mapas generando nuevos mapas y permitiendo buscar objetos que satisfagan ciertas condiciones. Esta capa será la capa más utilizada desde el punto de vista del usuario ya que, en general, el usuario interactúa con mapas donde puede visualizar los objetos ubicables de su interés. A nivel de construcción de modelos, resulta ser la capa más atractiva ya que nos permite especificar que objetos ubicables son lo que vamos a ubicar y luego mostrar la ubicación de cada objeto desde un sistema de representación particular.

5.5. Instanciación de la arquitectura

La arquitectura propuesta anteriormente fue instanciada en un framework con el objetivo de que sea validado mediante casos de uso. Como se enunció en la sección 5.2 el framework fue desarrollado sobre la distribución de Smalltalk: VisualWorks 7.4. A continuación se detallará como se ha concretizado cada una de las capas de la arquitectura en un conjunto de objetos que dan soporte para ubicaciones. Más adelante (Capítulo 6) se mostrará como se extendió el modelo del congreso para enriquecerlo con ubicaciones en la aplicación propuesta.

5.5.1. Objetos ubicables

La capa de objetos ubicables intenta agregar el comportamiento y el conocimiento necesario a los objetos del dominio respecto de sus ubicaciones. Esto motivo el modelado de la clase *LocatedObject* (Figura 5.3) con el objetivo de representar a un objeto ubicable. Debido a que cualquier objeto puede ser un objeto ubicable, la clase *LocatedObject* fue modelada como un Wrapper [13] genérico el cual puede envolver a cualquier objeto. Gracias a las bondades de un lenguaje de objetos puro como Smalltalk, cualquier mensaje que el Wrapper recibe y no entiende es automáticamente delegado al objeto que se encuentra wrapeando. Para hacer que un objeto como un aula sea un objeto ubicable debemos enviarle el mensaje *asLocatedObject*. El resultado del envío de dicho mensaje es el wrapper anteriormente enunciado.

En general los objetos ubicables pueden ser de dos tipos: estáticos o dinámicos. Los primeros tienen la propiedad de que su ubicación no cambia con el

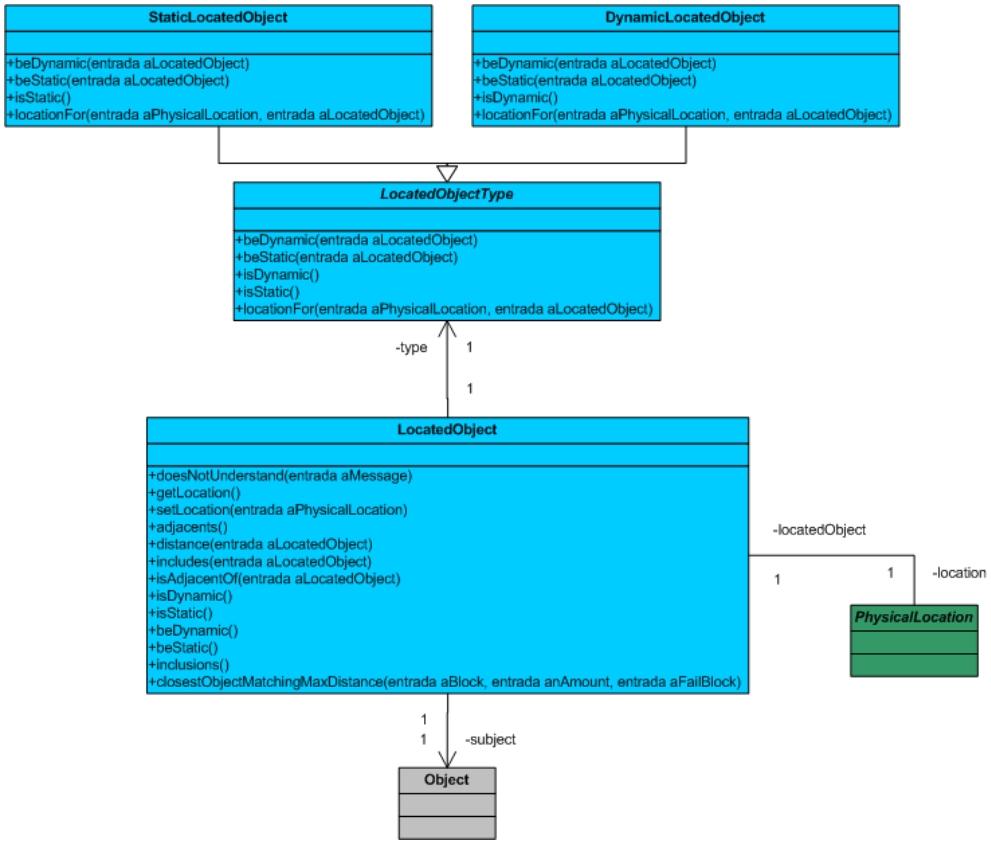


Figura 5.3: Instanciación de capa de objetos ubicables

tiempo, mientras que los segundos pueden cambiar su ubicación. Sin embargo el carácter de estático o dinámico es subjetivo. Un objeto puede ser estático durante un determinado tiempo y luego ser dinámico. Esto provoca que un objeto pueda cambiar su tipo a través del tiempo. Los tipos de objetos ubicables son considerados en el framework. Para ello se implementó el tipo de objeto ubicable mediante la clase *LocatedObjectType* utilizando el patrón Type Object [21]. Cuando un objeto estático se le intenta cambiar su ubicación enviando el mensaje *location:* provoca un fallo debido a la propiedad enunciada anteriormente. El tipo de un objeto ubicable puede ser cambiado enviando los mensajes *beStatic* o *beDynamic* a un objeto ubicable.

Por otro lado, un objeto ubicable expone un protocolo interesante para ver como este se relaciona con otros objetos ubicables. Por ejemplo un *LocatedObject* entiende los mensajes:

- *adjacents*: retorna el conjunto de *LocatedObject* que son adyacentes.

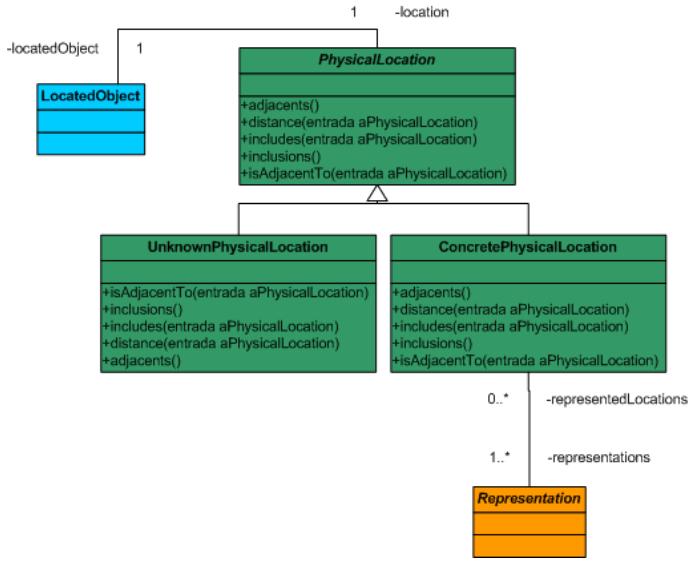


Figura 5.4: Instanciación de capa de ubicaciones

- *containers*: retorna el conjunto de *LocatedObject* en los cuales se encuentra incluido.

- *closestObjectMatching: aBlock maxDistance: anAmount ifNone: aFailBlock*: que retorna el objeto más cercano con una cota de distancia que satisfaga una condición. En caso de fallar se ejecuta el bloque *aFailBlock*.

5.5.2. Ubicaciones

Las ubicaciones físicas de los objetos ubicables fueron modeladas mediante la clase *PhysicalLocation* (Figura 5.4). En este caso hemos distinguido entre dos ubicaciones particulares. Una ubicación física concreta (*ConcretePhysicalLocation*) que nos permite establecer en base a representaciones la ubicación y una ubicación desconocida *UnknownPhysicalLocation*. Esta última sigue el patrón de diseño Null Object [42] proveyendo un objeto que representa una ubicación sin representaciones.

En base a lo relevado en el capítulo 2, la ubicación nos debe permitir computar las tres operaciones primitivas: distancia, inclusión y adyacencia. Para ello, utilizando la técnica del Double dispatching, las operaciones son resueltas a nivel de ubicación primero y luego en el caso de que sean dos ubicaciones concretas podremos computar la operación a través de sus representaciones. Es decir podríamos computar la distancia entre una ubicación concreta y una ubicación desconocida obteniendo un valor nulo para tal caso.

Las ubicaciones concretas establecen la semántica de sus operaciones en base a sus representaciones. En el capítulo 4 habíamos anticipado que ocurriría cuando algunas representaciones afirmen el resultado de una operación y otras lo nieguen. Para ello hemos definido diferentes políticas para el cómputo de las operaciones entre ubicaciones concretas. Esto sigue la idea de una persona realizando varios ensayos de los cuales obtiene diferentes resultados. Luego es necesario decidir cual es el resultado del ensayo general. Para ello se definieron diferentes políticas dependiendo del tipo de operación a realizar. Dichas políticas fueron implementadas siguiendo el patrón de diseño Policy [13] en donde se desacopla la implementación de una operación en una jerarquía de clases.

Las políticas desarrolladas para las operaciones donde la respuesta es verdadero o falso son las siguientes:

- *AnySatisfyPolicy*: Devuelve verdadero en el caso que al menos un par de representaciones dijo que si.
- *AllSatisfyPolicy*: Devuelve verdadero si todos los pares de representaciones contestaron que si.
- *PorcentageSatisfyPolicy*: Devuelve verdadero en el caso de que más de un determinado porcentaje de pares de representaciones haya respondido en forma afirmativa.

Por su parte la operación de distancia puede devolver diferentes resultados que son necesarios compatibilizar. Por dicho motivo se han desarrollado las siguientes políticas:

- *MaximumPolicy*: Devuelve el máximo entre todas las distancias computadas por las representaciones.
- *MinimumPolicy*: Devuelve el mínimo entre todas las distancias computadas por las representaciones.
- *AveragePolicy*: Devuelve un promedio entre todas las distancias computadas por las representaciones.
- *WisePolicy*: Toma la media entre todas las distancias y en base a una varianza configurada devuelve el promedio entre las distancias que tengan media: la media calculada y estén dentro de la varianza configurada.

Para cada una de las operaciones se han configurado políticas por default con el objetivo de que el desarrollador no deba lidiar con la responsabilidad de decidir cual de ellas utilizar. Dicha elección fue realizada en base a la utilización del framework en el desarrollo del caso de uso. Para el caso de las operaciones de adyacencia e inclusión se optó como política por default a: *AnySatisfyPolicy*. Por

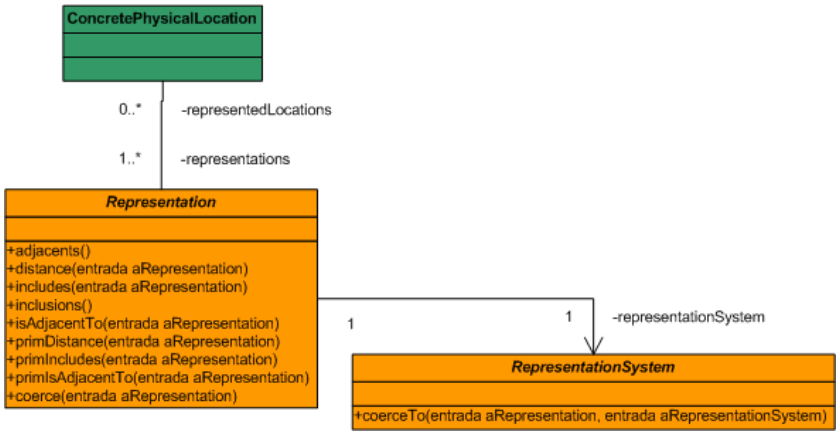


Figura 5.5: Instanciación de capa de representaciones

otro lado en el caso de la operación de distancia se optó la política *WisePolicy* con el objetivo de establecer un consenso entre todas las distancias calculadas por las representaciones.

5.5.3. Representaciones

Continuando con las ideas propuestas en el capítulo 4 habíamos introducido el concepto de representación con el objeto de diferenciar el concepto abstracto de ubicación de como esta era representada en un modelo. Los modelos son ahora los sistemas de representación que permiten establecer el contexto donde una representación es válida. Por dicho motivo la capa de representaciones contemplará estas dos clases abstractas: *Representation* y *RepresentationSystem* (Figura 5.5).

La clase *Representation* modela en forma abstracta lo que significa representar ubicaciones. Para ello cada representación conoce el conjunto de ubicaciones que representa. Cuando una ubicación concreta es representada mediante una representación es ahí cuando la relación entre la ubicación y la representación son conectadas.

Las representaciones deben operar entre ellas cuando son compatibles, es decir cuando pertenecen a un sistema de representación en común. Por dicho motivo las representaciones son coercionadas en el momento que invocamos un mensaje como puede ser la inclusión. En caso que la coerción no pueda ser realizada se lanza una excepción con el fin de que sea la operación la que decida que resultado devolver. En caso contrario las representaciones podrán ser convertidas a un sistema de representación en común y es en este punto donde la operación puede ser resuelta.

La representación sigue el patrón de diseño Template Method [13] debido

a que la representación define el esqueleto para el cómputo de la operación coercionando el operando en primera instancia y dejando que cada subclase implemente la operación en el sistema de representación particular.

La clase *RepresentationSystem* modela el sistema de representación abstracto. El sistema de representación define el contexto donde una representación es válida. Por dicho motivo es quien será responsable de instanciar las representaciones. De esta forma el sistema de representación jugará el papel de factory de representaciones en la capa de representaciones. Cada una de las subclases de *RepresentationSystem* deberán publicar en su protocolo un conjunto de mensajes que permitan instanciar a las representaciones en dicho sistema.

5.5.4. Representaciones geométricas

La representación abstracta provee un esqueleto para definir representaciones para las ubicaciones. Para poder utilizar las representaciones necesitamos concretizar las clases abstractas en clases concretas que permitan representar realmente a las ubicaciones. Uno de los modelos que habíamos visto eran los modelos geométricos (Capítulo 2) los cuales permiten representar a la ubicación como un punto, un polígono o una polilínea. Por dicho motivo el sistema de representación geométrico proveerá de tres representaciones (Figura 5.6).

Habíamos visto que los modelos geométricos podían ser implementados de tres formas diferentes: un raster, un spaghetti o un modelo topológico. A nivel conceptual, siempre nos encontramos trabajando con las mismas tres abstracciones: punto, polilínea y polígono las cuales se encuentran implementadas de formas diferentes. Esto provocó la inclusión de las clases *GeometricPlanarImplementation* y *PlanarModel*. La primera es una clase abstracta para las implementaciones de cada una de las abstracciones. Entre la representación geométrica y su implementación surge el patrón Bridge [13] que nos permite desacoplar una abstracción de su implementación. Cada uno de los modelos (raster, spaghetti y topológico) son definidos subclasificando la clase *PlanarModel*. Los modelos son los responsables de construir las implementaciones de cada una de las representaciones. En este caso aplicamos el patrón Abstract Factory [13] el cual nos permite crear una implementación de cada una de las primitivas dependiendo del modelo que estemos utilizando.

Los modelos geométricos pueden relacionarse mediante transformaciones que especifiquen como transformar un elemento de un conjunto en un elemento de otro conjunto. Las transformaciones nos permiten transformar una representación que se encuentra en un modelo geométrico en otra representación equivalente en otro sistema de representación. Dentro de las transformaciones implementadas (Figura 5.7) encontramos:

- *NullTransformation*: El sistema de representación no puede ser transformado en otro.

- *ConcreteTransformation*: Nos permite transformar las representaciones del sistema en representación de otro sistema mediante una o varias transformaciones de puntos.

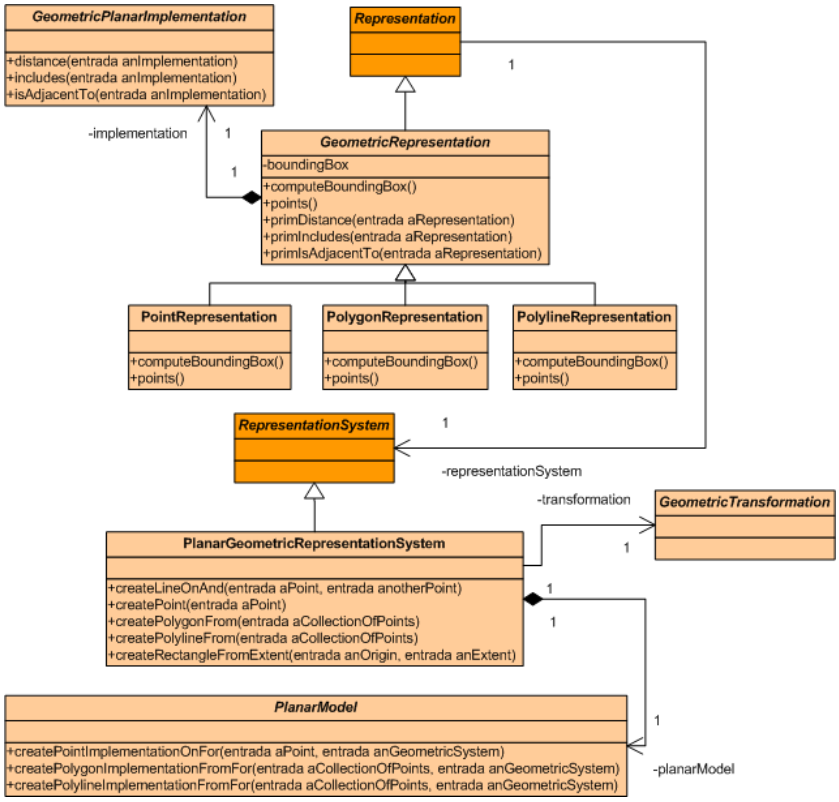


Figura 5.6: Instanciación de capa de representaciones geométricas

Las transformaciones de puntos utilizadas son las más comunes (Figura 5.7):

- *RotationTransformation*: Permite rotar los puntos mediante una medida angular.

- *ScaleTransformation*: Permite escalar los puntos mediante un factor de escala.

- *TranslationTransformation*: Permite trasladar los puntos mediante una distancia en x y una distancia en y.

Para indicar que un sistema es una transformación de otro debemos enviar una o varias veces los siguientes mensajes:

- *rotatedBy: aPoint from: aRepresentationSystem*: Nos permite indicar que el sistema es una rotacion de otro sistema.

- *scaledBy: aPoint from: aRepresentationSystem*: Nos permite indicar que el sistema es una escala de otro sistema.

- *translatedBy: aPoint from: aRepresentationSystem*: Nos permite indicar que el sistema es una traslación de otro sistema.

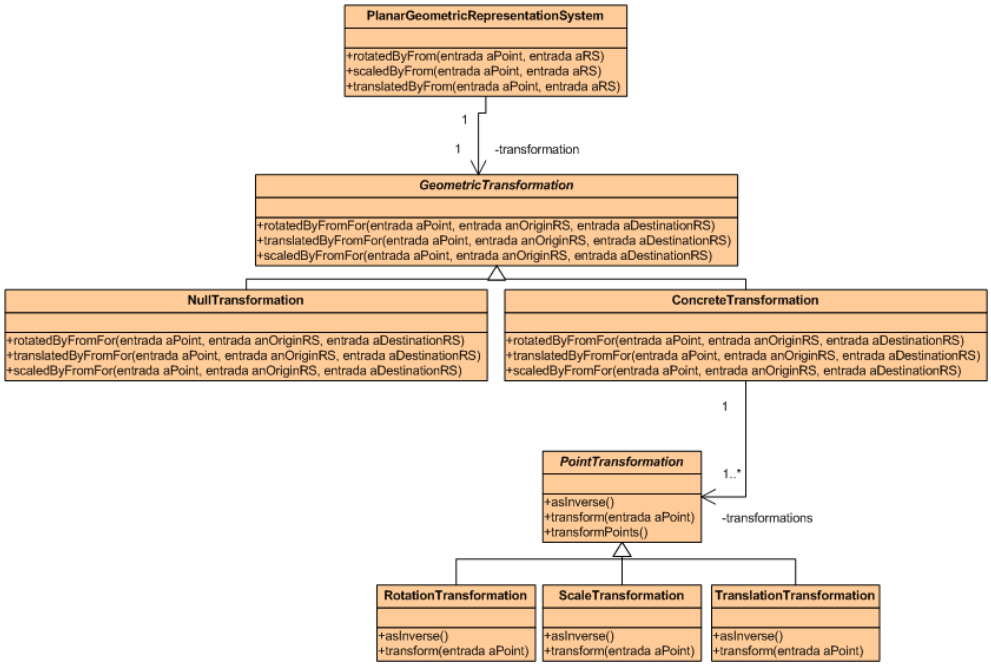


Figura 5.7: Transformaciones geométricas

Una operación entre dos representaciones geométricas es resuelta si y solo si ambas representaciones pueden ser coercionadas a un sistema de representación en común. Para ello la clase *GeometricRepresentation* coopearando con su sistema de representación intentan llegar a un sistema de representación común utilizando la transformación que posee el sistema. Una vez que las representaciones se encuentran en el mismo sistema de representación recién ahí se resuelve la operación. Para ello la representación geométrica delega el comportamiento de resolver la misma en la implementación que la misma posee en alguno de los modelos (raster, spaghetti, topológico).

5.5.5. Representaciones simbólicas

Las modelos simbólicos que habíamos visto en el capítulo 2 permiten representar a las ubicaciones como símbolos que se encuentran relacionados. Dichas relaciones son las que definen la semántica de cada una de las operaciones. Habíamos enunciado que las operaciones podían estar definidas por comprensión o por extensión sobre el conjunto de símbolos. En nuestro caso hemos definido el modelo simbólico por extensión el cual es lo suficientemente general como para poder ser utilizado en una variedad de aplicaciones.

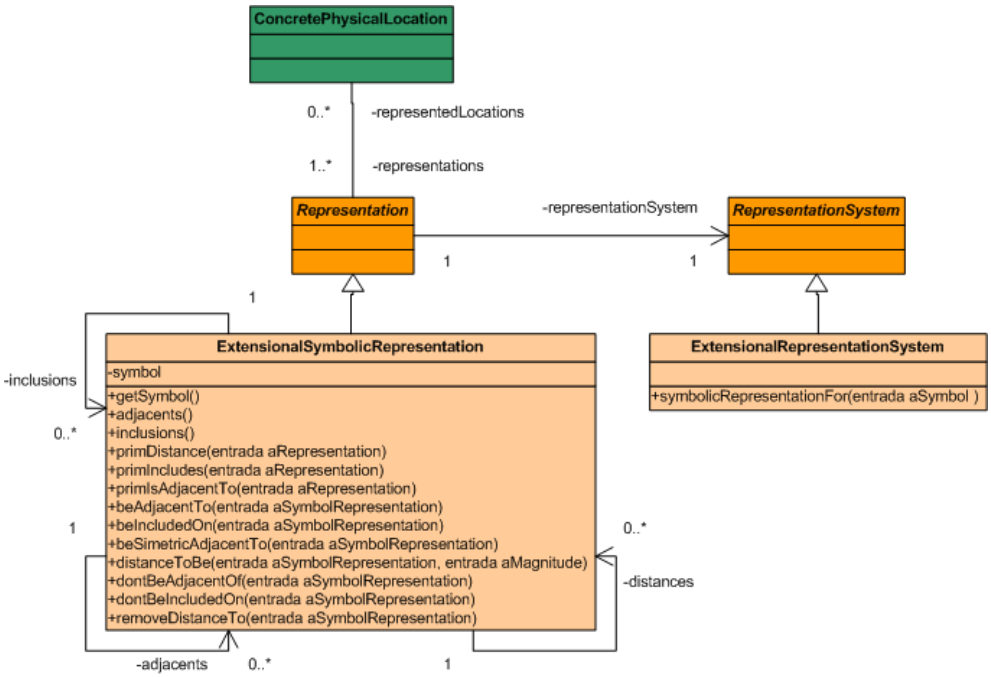


Figura 5.8: Instanciación de representaciones simbólicas

Para implementar los modelos simbólicos subclasificamos la clase Representation (Figura 5.8) con la clase *ExtensionalSymbolicRepresentation* la cual define por extensión el conjunto de símbolos con los cuales se encuentra relacionada. Dicha clase permite definir las relaciones con otros símbolos mediante los siguientes mensajes:

- *beAdjacentOf: aSymbolicRepresentation*
- *beIncludedOn: aSymbolicRepresentation*
- *beSimetricAdjacentOf: aSymbolicRepresentation*
- *distanceTo: aSymbolicRepresentation be: aMagnitude*

- *dontBeAdjacentOf: aSymbolicRepresentation*
- *dontBeIncludedOn: aSymbolicRepresentation*
- *removeDistanceTo: aSymbolicRepresentation*

La implementación utiliza relaciones de conocimiento para implementar las relaciones de inclusión y adyacencia entre símbolos. Para el caso de la operación de distancia se posee un diccionario que mapea los pares (representación, distancia).

El sistema de representación, modelado por la clase *ExtensionalRepresentationSystem*, permite crear representaciones mediante el mensaje *symbolicRepresentationFor: aSymbol*. Dada la capacidad de extensión del ambiente elegido, podemos utilizar el mecanismo de extensión de clases para la clase *String* entienda el mensaje *asRepresentationOn: aRepresentationSystem*. El mismo se encuentra implementado de la siguiente forma:

```
String>>asRepresentationOn: aSymbolicRepresentationSystem

        ^aSymbolicRepresentationSystem
              symbolicRepresentationFor: self asSymbol
```

5.5.6. Nuevas representaciones

Con el modelado de objetos del dominio ocurre que muchas veces las relaciones de conocimiento entre dos objetos tienen cierta semántica oculta con respecto a las ubicaciones. Volviendo al ejemplo del sistema para el congreso, observemos la relación que existe entre las aulas y los edificios. Dicha relación nos esta expresando también una relación de inclusión entre el edificio y las aulas. Es decir un Building tiene muchas Room y una Room pertenece a un único Building.

Con el objetivo de aprovechar estas relaciones físicas que surgen del modelado con objetos se agrego una nueva forma de representar ubicaciones, las representaciones dependientes del dominio. Para ello definimos una clase DomainRepresentationSystem (Figura 5.9) que nos permite definir como obtener los objetos adyacentes, como obtener en cuales objetos se encuentra incluido y las distancias con respecto al resto de los objetos. Dicha configuración es realizada mediante tres bloques. Una vez que hemos instanciado nuestro nuevo sistema de representación podemos instanciar representaciones dentro de ese sistema enviando el mensaje *asRepresentationOn:* a los objetos de interés.

Como ejemplo tomemos el caso del edificio y las aulas. Definamos un nuevo sistema de representación basado en este modelo mediante el siguiente código:

```
rs := DomainRepresentationSystem
```

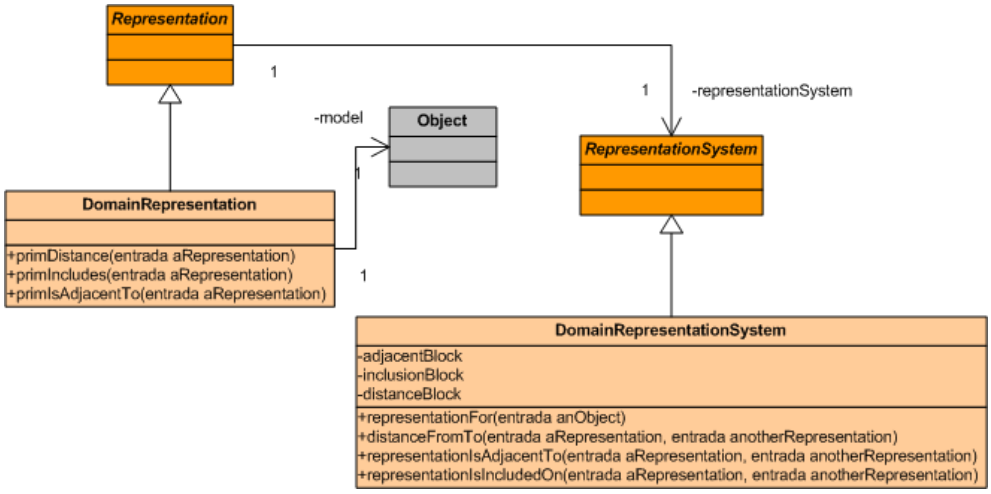


Figura 5.9: Representaciones dependientes del dominio

```
    inclusionWith:[:object |
                    object accept: InclusionsFor new ].
```

La introducción de la clase InclusionsFor, siguiendo el patrón Visitor [13], nos permite ejecutar una operación sobre los elementos de una jerarquía sin que estos tengan que implementarlo expresamente. La clase *InclusionsFor* implementa los visit de la siguiente forma:

```
InclusionsFor>>visitBuilding: aBuilding

    ^#()

InclusionsFor>>visitRoom: aRoom

    ^Array with: aRoom building
```

Una vez que hemos definido el sistema de representación debemos agregar tanto a las salas como a los edificios sus correspondientes representaciones. Para ello crearemos una nueva ubicación en base a la actual de la siguiente forma:

```
buildings do:[:building |
  buildingRepresentation := building asRepresentationOn: rs.
  newLocation := building location
                  copyWith: buildingRepresentation.
  building location: newLocation.
].
```

```
rooms do:[:room |
  roomRepresentation := room asRepresentationOn: rs.
  newLocation := room location
                  copyWith: roomRepresentation.
  room location: newLocation.
].
```

De esta forma utilizamos el propio objeto de dominio como representación de la ubicación, aprovechando relaciones del modelo de objetos podemos agregar contenido semántica a la ubicación.

5.5.7. Mapas abstractos y concretos

Los mapas nos permiten agrupar un conjunto de objetos ubicables con el objetivo de mostrar sus relaciones. En el capítulo 4 habíamos introducido el concepto de mapa abstracto con el objetivo de referirnos a los mapas que no tomaban en cuenta ningún sistema de representación en particular. Cuando queríamos ver a dicho mapa en algún sistema en particular lo concretizábamos en dicho sistema de representación. En nuestro framework dichos conceptos son mapeados a las clases *AbstractMap* y *ConcreteMap* (Figura 5.10).

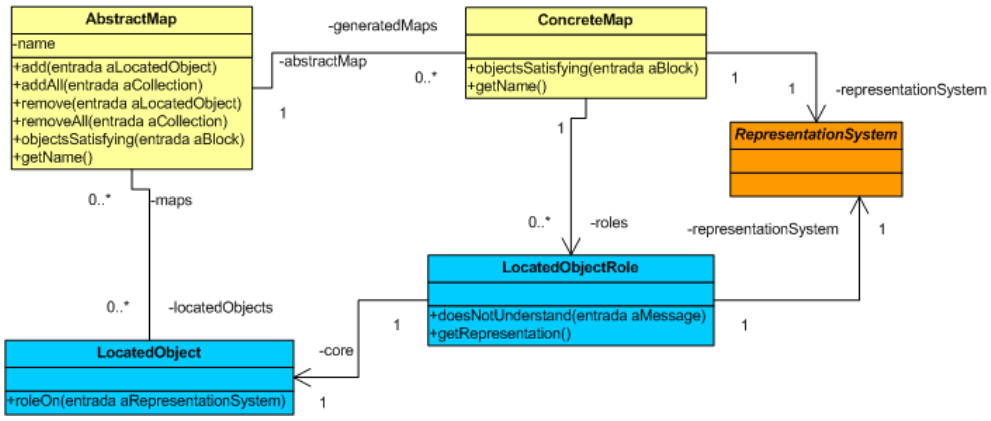


Figura 5.10: Instanciación de capa de mapas

Un *AbstractMap* representa un mapa abstracto el cual posee un nombre y agrupa un conjunto de objetos ubicables. El mapa permite buscar los objetos que satisfacen ciertas condiciones mediante el mensaje *objectsSatisfying: aBlock* el cual nos devuelve una colección de objetos que satisfacen el bloque *aBlock*. Un mapa permite el agregado y eliminación de objetos mediante los mensajes: *add:*, *addAll:*, *remove:* y *removeAll:*. Como hemos dicho un mapa abstracto puede ser concretizado en un sistema de representación en particular mediante el mensaje *concreteMapOn: aRepresentationSystem*.

Los mapas concretos son modelados mediante la clase *ConcreteMap*. Dicha clase nos permite ver a los objetos ubicables en un sistema de representación en particular. Por dicho motivo el mapa concreto conoce al sistema de representación en el cual se encuentra concretizado. La idea de concretizar un mapa promueve la idea de trabajar con los objetos ubicables desde un punto de vista del sistema de representación en el cual se ha concretizado. Por dicho motivo se introdujo la clase *LocatedObjectRole* la cual nos permite modelar el rol (patrón de diseño Role Object [1]) que cumple el objeto ubicable en el sistema de representación. Por ejemplo la ubicación de un aula se encuentra representada mediante un rectángulo en el sistema de representación geométrico de la facultad. Por dicho motivo el rol que cumple dicho objeto ubicable en el sistema de representación es el de un polígono. Si al rol le preguntamos cual es su bounding box este responderá lo mismo que si le hubiéramos preguntado a la representación.

Como hemos visto los objetos ubicables pueden estar incluidos en varios mapas. Conocer a que mapas pertenece un objeto ubicable nos permite visualizarlo desde diferentes puntos de vista. Cada mapa nos provee una visión particular del universo de objetos ubicables, por lo cual saber en que mapas se encuentre un objeto ubicable resulta interesante a nivel de usuario. Por dicho motivo la relación de conocimiento entre los mapas abstractos y los objetos ubicables es hacia ambos lados con el objetivo de que saber en que mapas puedo ver a un objeto no requiera una búsqueda particular.

5.5.8. Cómputo de caminos

El cómputo de caminos es uno de los aspectos más interesantes del trabajo con ubicaciones. Dar un camino desde un objeto ubicable hacia otro nos permite guiar a la persona que utiliza la aplicación para recorrer el espacio. Aunque el objetivo principal del presente trabajo es desarrollar una arquitectura que permita escalar e integrar los modelos de ubicación, el cómputo de caminos no debe ser un aspecto que deba ser dejado de lado.

El trabajo realizado sobre cómputo de caminos toma en cuenta las siguientes consideraciones:

- La utilización de algoritmos como Dijkstra [7] son imposibles de ser utilizados en grafos con millones de nodos (objetos ubicables).

- Del punto anterior se concluye que es necesario utilizar algoritmos que hagan uso de heurísticas como el algoritmo A* [24]. La heurística es algo variable que podrá modificar el usuario como puede ser: no utilizar calles que posean semáforos, usar avenidas, etc.

Guías

La elección de una heurística adecuada en el cómputo de caminos es una de las tareas más complejas de realizar. La elección de una buena heurística es lo que determina el éxito en el cómputo del camino. Los algoritmos heurísticos utilizan una única heurística a ser aplicada sobre cada uno de los nodos del grafo. Esto motivo a pensar que ocurre en la realidad cuando realizamos un viaje a un lugar desconocido. A medida que nos movemos físicamente, acostumbramos realizar paradas y consultar a los lugareños con el fin de que sean estos quienes nos guíen en el camino. Se asume que una persona del lugar tiene mayor conocimiento de cómo llegar a un lugar de interés, que el conocimiento que podemos poseer nosotros. Este hecho motivo la introducción del concepto de guía.

Un guía será quien nos dará la heurística de un nodo con respecto al destino que deseamos alcanzar. En nuestro modelo cada *LocatedObject* (Figura 5.11) conocerá a un *Guider* que hará de guía a la hora de tener que decidir entre varios caminos. La ventaja de este acercamiento es que el *Guider* puede variar entre los diferentes objetos ubicables. De esta forma podremos definir un guía para el contexto del edificio de la Facultad de Informática y otro guía para las calles de la ciudad. Cada uno de ellos será configurado por separado pero cooperaran cuando deseemos computar un camino desde un objeto ubicable que se encuentra en un edificio hasta otro objeto ubicable en otro edificio.

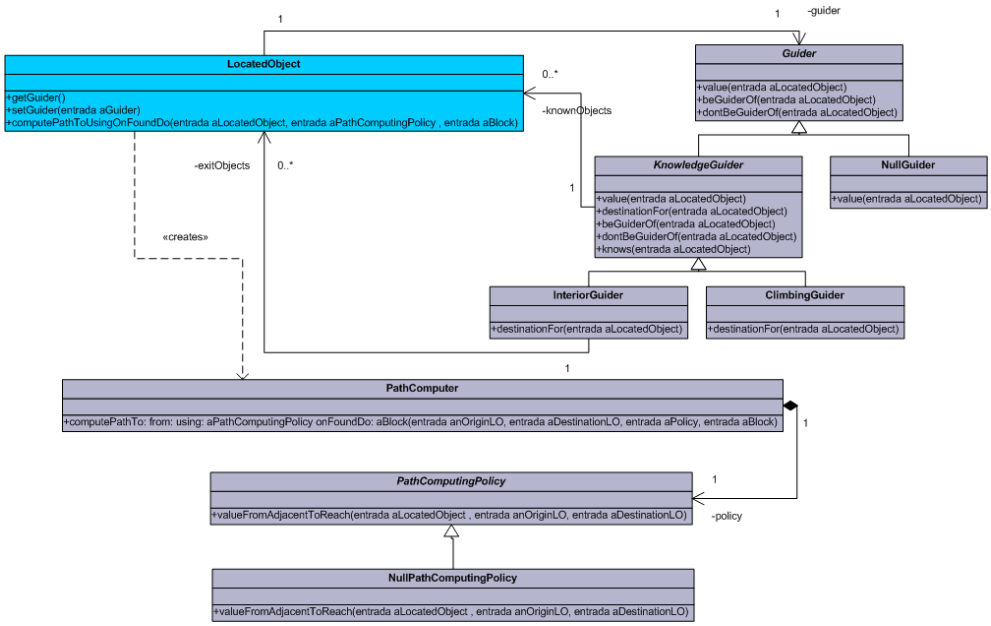


Figura 5.11: Cómputo de caminos

El guía quedo concretizado en la clase abstracta *Guider*. Cada subclase

de *Guider* deberá implementar el mensaje *value: aLocatedObject fromAdjacent: anOriginLocatedObject toReach: aDestinationLocatedObject using: aPathComputingPolicy*. El objetivo es dar un valor heurístico del objeto *aLocatedObject* el cual fue alcanzado desde *anOriginLocatedObject* con el objetivo de llegar a *aDestinationLocatedObject*. La idea es que el *Guider* colabore con la política de computo de caminos con el fin de obtener la heurística final. La política para el cómputo de caminos, es un objeto que puede configurar el usuario dependiendo del dominio de la aplicación. Puede ser tan variable como porque objetos prefiere pasar, cuales quiere evitar, etc. Es decir, la política valúa la heurística desde un punto de vista del dominio de la aplicación. El guía valúa la heurística desde un punto de vista físico.

Con el desarrollo del framework y del caso de uso se han identificado tres guías diferentes:

- *NullGuider*: es un guía que provee una implementación por default para guiar en base a la distancia que existe entre el objeto *aLocatedObject* y el objeto destino *aDestinationLocatedObject*.

- *InteriorGuider*: es un guía que es común que sea utilizado en objetos ubicables de edificios. Este guía conoce a un conjunto de objetos internos y un subconjunto de ellos que representan puntos de salida. Lo que realiza el guía es: en caso de conocer al destino guiarlo así allí en base a la distancia existente. En caso contrario, lo guía hacia la salida más cercana.

- *ClimbingGuider*: es un guía que es utilizado comúnmente en lugares de exteriores. Este guía conoce a un conjunto de objetos. Cuando necesita dar una heurística para un objeto que conoce calcula dicha heurística en base a la distancia física que existe entre ellos. En caso contrario, intenta darle una heurística a alguno de los objetos en los cuales se encuentra incluido dicho objeto. Por ejemplo, si me encuentro en las calles de la ciudad y necesito ir hacia un aula no puedo establecer una heurística para el aula ya que es un objeto que no conozco, pero si puedo establecer una heurística para el edificio que se encuentra en el mapa de la ciudad.

La clase *LocatedObject* fue extendida agregando el mensaje *computePathTo: aLocatedObject using: aPathComputingPolicy onFoundDo: aBlock*. De esta forma podemos solicitar un camino desde una *Room* que se encuentra en un edificio hasta otro objeto ubicable que se encuentra en otro edificio. El camino computado permitirá indicar como salir del primer edificio, recorrer las calles de la ciudad hasta llegar al otro edificio, ingresar dentro de dicho edificio y completar el camino en este ultimo edificio. A lo largo del camino se utilizan tres *Guider* distintos. Dos *Guider* de interior para el caso de los edificios y un *ClimbingGuider* para el caso de las calles.

Configuración de los objetos ubicables

Una vez que hemos definido que guía utilizaran los objetos ubicables será necesario configurar dichos objetos con su guía correspondiente. Para ello deberemos instanciar el guía en cuestión y configurar los objetos ubicables de la siguiente forma:

```
guider := InteriorGuider new.
rooms do:[:room | room guider: guider].
```

Algoritmo para cómputos de caminos

A continuación se detallará la implementación del algoritmo para el computo de caminos el cual se basa en el algoritmo A*.

1. Iniciacion:
   a) Sea *origen* el objeto ubicable origen del camino y *destino* el destino del camino.
   b) F el conjunto de objetos ubicables frontera. Inicialmente $F = \{x/x \in origen\ adjacents\}$.
   c) h la función heurística definida en base al guía del objeto ubicable recibido como argumento.
   d) Sea V el conjunto de objetos visitados. Inicialmente $V = \{origen\}$.
   e) Sea $A = o'$ tal que $h(o')$ es el mínimo del conjunto $\{x/x = h(o)$ y $o \in F\}$.
2. Iteracion:
   a) Si $A = destino$ entonces termino.
   b) Si $F = vacio$ entonces termino no hay camino.
   c) Sino $F = F \cup \{o/o \in A\ adjacents$ y $o \notin V\}$.
   d) $V = V \cup \{A\}$.
   e) Recomputar A como antes.

Capítulo 6

Caso de prueba

6.1. Dominio de la aplicación

Con el objetivo de darle validez a la arquitectura desarrollada se la instanció en un framework que permite el desarrollo de aplicaciones que hagan uso de la ubicación. Dicho framework fue utilizado en el desarrollo de una aplicación que ofrece distintos servicios basados en la ubicación. La misma es similar a la propuesta por Valerie Bennett y Andrew Capella [38] y esta basada en una extensión de la aplicación del congreso.

La aplicación (Figura 6.1) permitirá a los usuarios estar ubicados en un conjunto de lugares donde se desarrollara el congreso. Estos lugares son los edificios de la facultad de informática de la UNLP y en la catedral de la ciudad. El objetivo es que las personas que asisten al congreso de la ciudad puedan guiarse tanto en los lugares donde el congreso se llevara a cabo como en un lugar turístico como es la catedral.

Para el caso del propio congreso se ha desarrollado un modelo que permite saber en que horario se realizaran las presentaciones de las publicaciones. Se dispone además de los abstract y títulos de los artículos de cada expositor realizará con el objetivo de que las personas que estén escuchando al orador puedan visualizar por escrito parte de la publicación en cuestión. En el caso de la catedral se desea visualizar un mapa que permita orientar a las personas que la visiten. Por otro lado se proveerán de un listado de sitios de interés como restaurantes y lugares de alojamiento con el objetivo de que en cualquier momento la aplicación les permita ubicarse geográficamente y calcular caminos hacia ellos.

Debido a que no se dispone de la información cartográfica necesaria para computar caminos correctamente, no se podrán computar caminos diferenciando si el recorrido se realiza caminando o a través de las calles de la ciudad. También se excluirán los comentarios referentes a la aplicación desarrollada pa-

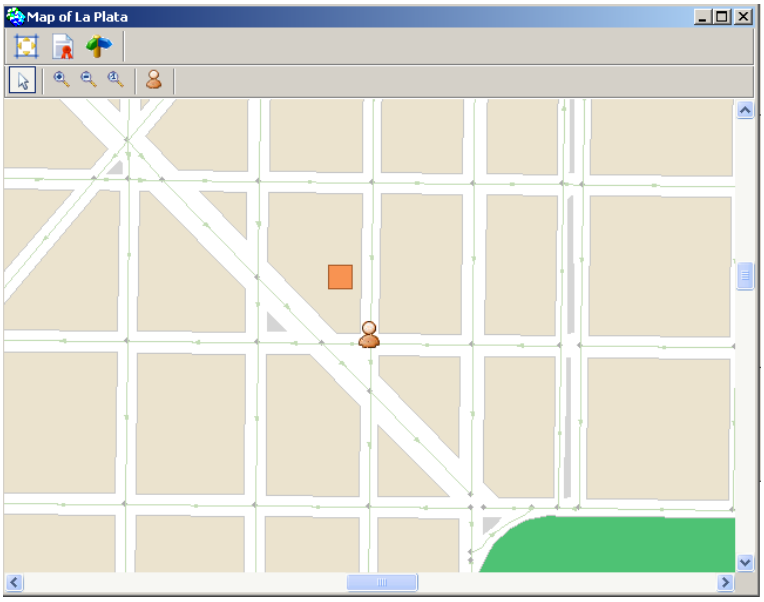


Figura 6.1: Aplicación

ra el congreso debido a que el foco del trabajo es el manejo de las ubicaciones.

6.2. Desarrollo de modelos

Cada uno de los modelos de ubicación fueron desarrollados por separado con el objetivo de que cada uno de ellos sea integrado en una etapa posterior. Para comenzar se desarrollo el modelo de ubicaciones para la ciudad de la plata a partir de un conjunto de Shape files [12]. Este tipo de archivos aloja un conjunto de representaciones geométricas las cuales se encuentran enlazadas con una base de datos en formato DBF que posee información del dominio. Para construir el modelo de las calles de la ciudad se leen varios shape files que representan un conjunto de objetos ubicables de interés. Dentro de este conjunto de objetos ubicables podemos destacar a: los segmentos de calles, las manzanas, los bulevares de la ciudad y las plazas. Sobre el mapa de la ciudad (Figura 6.2) se incluyeron a los restaurantes y los lugares de alojamiento para que las personas que participen del congreso puedan estar ubicadas.

El siguiente paso fue desarrollar los modelos de ubicaciones de cada uno de los edificios de las facultades. Para ello se instanciaron cada una de las representaciones geométricas y simbólicas (Figura 6.3) que luego formaran las ubicaciones de dichos objetos. Este trabajo requirió una tarea manual que hubiera sido fácilmente resuelta si se hubiera desarrollado un editor.

Por ultimo, se desarrollo el modelo de ubicaciones de la catedral de la plata.

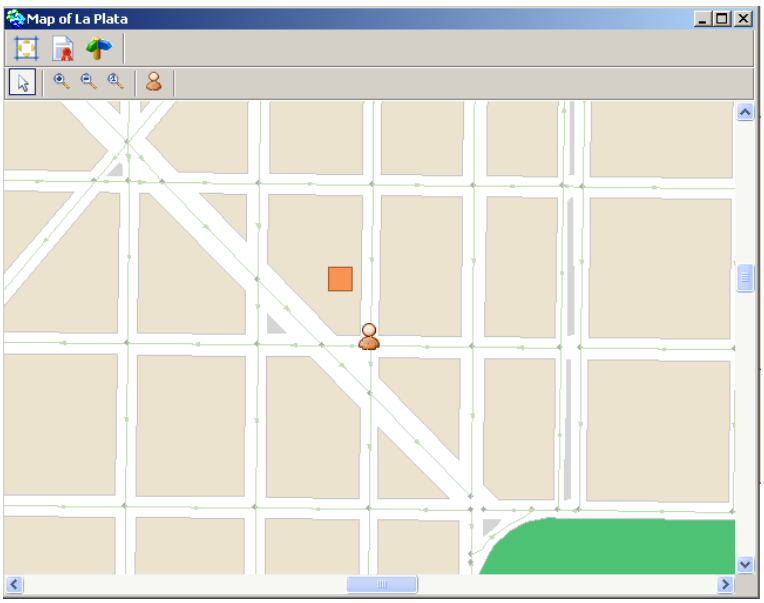


Figura 6.2: Mapa de la ciudad

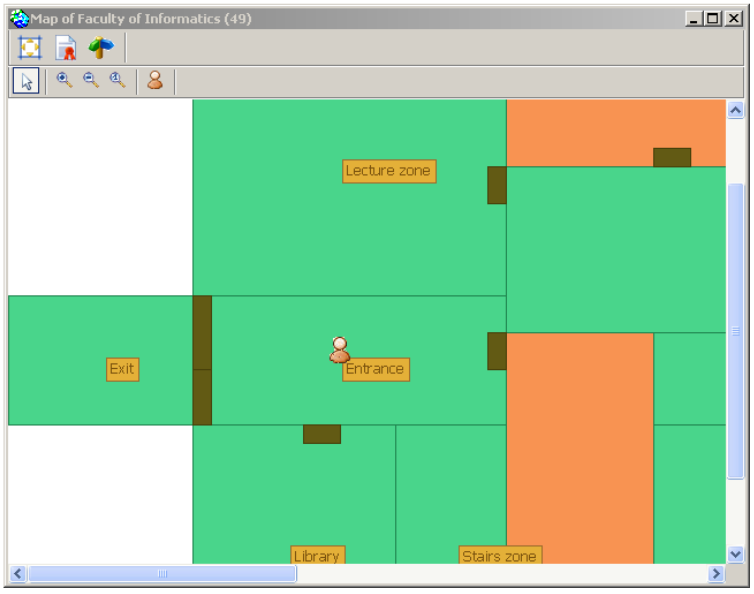


Figura 6.3: Modelo de la facultad 49

Delimitando zonas de interés dentro de la catedral y la ubicación del cristo entre otros (Figura 6.4).

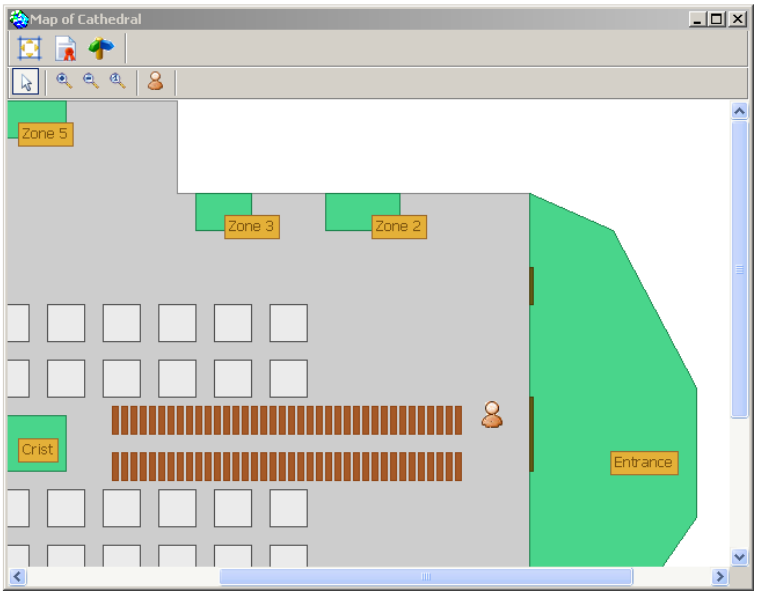


Figura 6.4: Modelo de la catedral

6.3. Integración de modelos

Una vez que hemos desarrollado cada uno de los modelos es necesario que los mismos sean integrados. Por ejemplo para el caso del modelo desarrollado para la catedral de la plata es necesario especificar porque lugares podemos ingresar y como estos se encuentran relacionados con las calles de la ciudad de la plata. Algo similar ocurre con los ingresos a los edificios de la facultad de informática. Por dicho motivo es necesario especificar las relaciones que existen entre objetos pertenecientes a un mapa con respecto a los objetos pertenecientes a otro.

En el caso de la catedral de la plata, se han especificado un punto de acceso. El mismo se encuentra enfrente a la plaza. Por dicho motivo hemos especificado que el segmento de calle que pasa por enfrente a la plaza es adyacente a la entrada de la catedral. Dicha tarea fue realizada agregando una representación simbólica que nos permite establecer la relación de la siguiente forma:

```
rs := ExtensionalRepresentationSystem new.

entradaPrincipalRep := #entradaPrincipal asRepresentationOn: rs.
callePlazaRep := #callePlaza asRepresentationOn: rs.
```

```
entradaPrincipalRep beSimetricAdjacentOf: callePlazaRep.

entradaPrincipal
    location: (entradaPrincipal location
                              copyWith: entradaPrincipalRep).
callePlaza
    location: (callePlaza location
                              copyWith: callePlazaRep).
```

De esta forma hemos integrado a la catedral de la plata con el modelo de calles del mapa de la plata. En forma análoga hemos realizado con los ingresos a los edificios de calle 115 y 49 pertenecientes a la facultad de informática.

6.4. Extensión de las ubicaciones

Integrar los modelos anteriores nos permite definir las relaciones que tienen los objetos que se encuentran sobre los bordes de cada uno de los edificios. Muchas veces es necesario extender la ubicación de los objetos en otros sistemas de representación provistos por otros mapas. Por ejemplo sería interesante tener una representación geométrica de la catedral en el sistema de representación geométrico de la ciudad. Dicha tarea puede ser resuelta agregando una nueva representación a la ubicación de la catedral y una posterior inclusión en el mapa de la ciudad de la siguiente forma:

```
rectangleRepresentation
     := ((3281@3408 extent:1) expandedBy: 20)
              asRepresentationOn: laplataGeometric.
cathedral
    location: (cathedral location
                  copyWith: rectangleRepresentation).

mapLaPlata add: cathedral.
```

En forma análoga podemos realizar con las ubicaciones de cada uno de los edificios de la facultad de informática. El resultado de dicho proceso puede ser visualizado en la figura 6.5.

6.5. Servicios ofrecidos

La aplicación desarrollada (Figura 6.6) provee una UI que permite a los usuarios visualizar constantemente donde este se encuentra ubicado. A su vez se permiten utilizar los siguientes servicios:

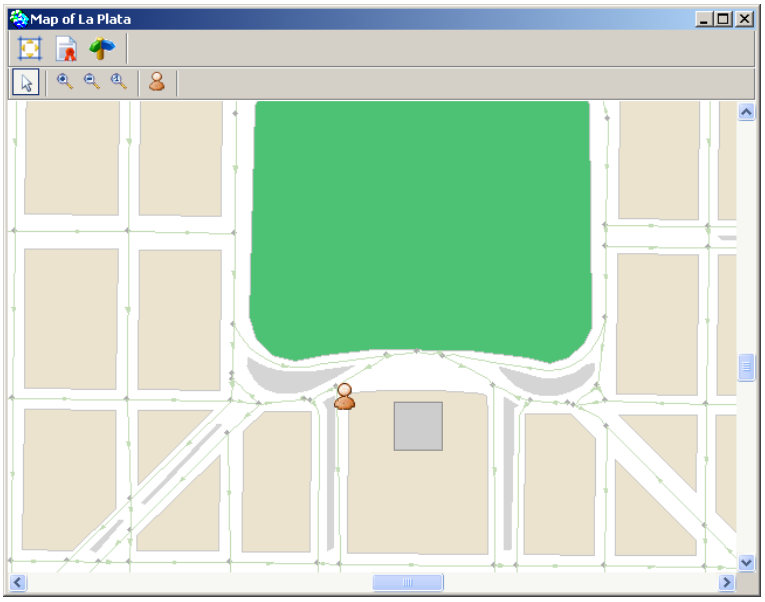


Figura 6.5: Integración de la Catedral

- Cómputo de caminos: Realizando botón derecho sobre un objeto ubicable podemos calcular un camino hacia él.
- Visualizar el material que esta siendo proyectado: Haciendo botón derecho sobre una sala podemos pedirle el listado de eventos que ocurrirán en dicha sala. De cada evento podemos ver el titulo, los autores y el abstract del trabajo.
- Buscar un lugar para almorzar o cenar: Ver un listado de los lugares recomendados para almorzar y alojarse. Para cada elemento de este listado podemos visualizarlo en el mapa y computar un camino hacia ellos.

6.6. Simulación

Para probar la aplicación desarrollada se desarrolló un sencillo entorno de simulación que permite simular la recepción de señales de dispositivos como GPS, Beacons y Kindergarten[1].

La utilización de dichos simuladores es sencilla. Solo se debe hacer click sobre las flechas en el caso del GPS y el Kindergarten para que la señal emitida cambie. Por otro lado para el caso de los beacons se provee un listado de los posibles símbolos a emitir, los cuales pueden ser cambiados seleccionándolos del listado.

[1] Tecnología en desarrollo para el posicionamiento en 2D en interiores

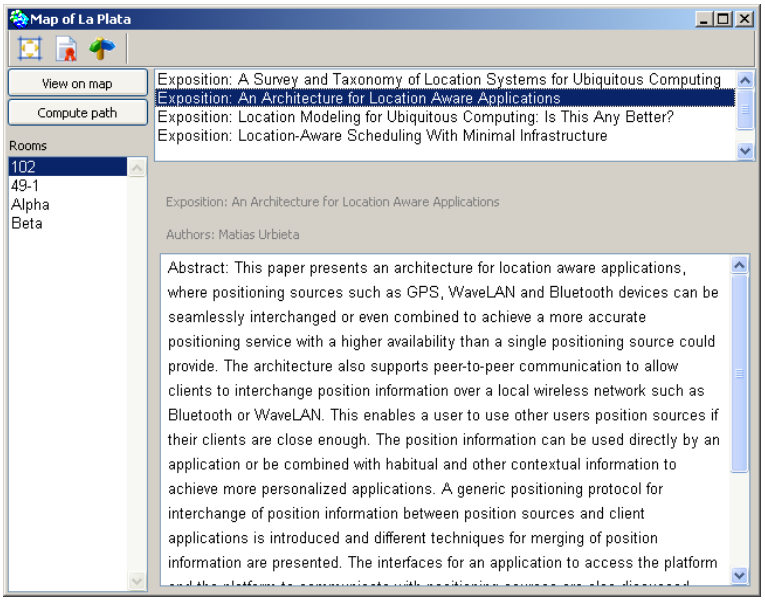


Figura 6.6: Aplicación de la conferencia

Capítulo 7

Guías de diseño

7.1. Guias

El desarrollo del framework para sistemas de ubicación y la implementación de un caso de uso provocan la adopción de ciertas metodologías a la hora de desarrollar modelos de ubicaciones. Dicho conocimiento puede ser transmitido a personas que no posean conocimientos informáticos con el objetivo de que ellas puedan construir, integrar y enriquecer sus modelos. Por dicho motivo el presente capítulo da un conjunto de guías sencillas que pueden ser utilizadas con el objetivo de desarrollar modelos de ubicaciones.

7.2. Definir que se intenta modelar

La primera guía es definir cual es el objeto de nuestro modelo. Las formas más sencillas de decidir que es lo que se intenta modelar es definir cual será el objeto más grande que contendrá el mapa. Por ejemplo deseo modelar las ubicaciones de la catedral de La Plata. De esta forma delimitamos los objetos que incluirá el mapa en términos de que objetos contiene la catedral.

Un aspecto importante cuando delimitamos que objetos serán incluidos es la granularidad de los mismos. Es decir con cuanto nivel de detalle especificaremos las relaciones entre los objetos. Por ejemplo, en el mapa de la catedral de la plata los bancos frente al altar pueden ser especificados con un alto nivel de detalle (uno por uno) o podemos determinar una zona que se llame bancos sin ondar en la cantidad de posee dicha zona.

7.3. Especificar que objetos pertenecerán al mapa

Luego de que conceptualmente se ha delimitado el mapa es hora de crear cada uno de los objetos que pertenecen al mapa. Aunque en cualquier momento se podrán agregar nuevos objetos al mapa es conveniente tener en claro que objetos agregaremos al mapa. En principio los objetos que sean de interés para el mapa serán agregados al mismo sin una ubicación conocida. El mapa hará de contenedor de estos objetos.

7.4. Modelar el objeto contenedor

Un aspecto que resulta adecuado, más aún si intentamos integrar y enriquecer los modelos, es modelar el objeto que contiene al resto. En nuestro caso modelaremos la catedral de la plata que contiene directa o indirectamente a todos los objetos pertenecientes al mapa. Esto nos beneficiará cuando queramos establecer la ubicación de una zona, por ejemplo, en el mapa de las calles de la plata. Al haber definido la ubicación del objeto que la contiene en ese sistema de representación estaremos definiendo en forma estimativa la ubicación de los objetos que contiene.

7.5. Elegir los sistemas de representación

Una vez que hemos definido que objetos contendrá el mapa necesitaremos definir en que sistemas de representación modelaremos las ubicaciones de cada uno de ellos. En general resulta más sencillo utilizar un modelo geométrico que permite visualizar de una forma sencilla las representaciones de las ubicaciones. Sin embargo, cuando necesitamos expresar relaciones de adyacencia resulta más sencillo hacerlo con un sistema de representación simbólico definido por extensión. Por ello, es conveniente en una primera instancia definir mediante un modelo geométrico las representaciones para cada uno de los objetos. Luego, en caso de ser necesario definir algunas relaciones, utilizar un sistema simbólico realizando dicha tarea manualmente.

7.6. Modelar primero, integrar y enriquecer después

En base a la experiencia recabada durante el desarrollo del caso de uso, podemos afirmar que es conveniente modelar el mapa con las ubicaciones de interés en una primera instancia. Esto facilita la tarea de concentrarse en el modelado del dominio de interés, delimitando que objetos serán incluidos en el mapa y como estos serán representados en los modelos de representación elegidos.

Una vez que se ha desarrollado el mapa, puede o no ser necesario integrar dicho mapa con otros mapas existentes. Esto puede provocar que surjan objetos que no habían sido considerados en primera instancia y que son necesarios para integrar los mapas. Por ejemplo, en el mapa de la catedral de la plata, las puertas no podrían haber sido tomadas en cuenta cuando se desarrollo el mapa de dicho lugar. Pero dichos objetos son necesarios si deseamos especificar como se relacionan las calles aledañas con las entradas a la catedral.

La integración del mapa de la catedral con el mapa de las calles de la ciudad es conveniente realizarlo con otros sistemas de representación diferentes. En general, las relaciones de adyacencia son las que importan por lo cual con un modelo simbólico nos alcanzará para extender el mapa original.

La integración provocará que al menos se enriquezca la ubicación del objeto que contiene al resto en alguno de los sistemas de representación provistos por el otro mapa. Por dicho motivo, una vez que hemos definido e integrado el mapa podremos agregarle a la catedral una nueva representación. En nuestro ejemplo, agregaremos una representación geométrica para la catedral con el objetivo de que la misma posea una representación en el sistema de representación geométrico de las calles.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

La incorporación de dispositivos móviles con capacidad de sensar su posición, permite abordar un nuevo conjunto de aplicaciones en donde la ubicación de los usuarios y objetos de interés toma principal atención. Dicho acontecimiento sucederá en los próximos años con la incorporación de dispositivos como GPS a los teléfonos celulares y la utilización de un ancho de banda mayor para las comunicaciones.

Las aplicaciones podrán estimar así sus ubicaciones respecto a otros objetos de interés tanto en los interiores de los edificios como así también en el exterior. Por ello, como se encuentre modelada la ubicación y que arquitectura de soporte para el trabajo con ellas es un aspecto de interés. En este trabajo de grado se presentó una arquitectura que permite dividir en capas cada uno de los aspectos necesarios para dar soporte al trabajo con ubicaciones. Se identificaron cada una de las capas en base a un análisis realizado previamente respecto de los términos utilizados al momento de modelar ubicaciones. Para ello se introdujo el concepto de representación con el objetivo de expresar como una ubicación se encuentra representada en un modelo (geométrico, simbólico, etc). Esto permite que una ubicación se encuentre múltiplemente representada permitiendo que el concepto de ubicación sea fácilmente escalable a otros modelos y que los distintos mapas presentados por los usuarios sean fácilmente integrables.

En base a la arquitectura desarrollada se instanció un framework con el objetivo de dar una validez practica al análisis teórico realizado. Para ello se instanció cada una de las capas en objetos que nos permiten expresar los conceptos introducidos en el capítulo 4. Dicho framework fue utilizado en una aplicación (Capítulo 6) que permite a las personas de un congreso ver las publicaciones de los eventos que se están desarrollando, ser guiados desde un lugar hacia otro

lugar y poder visualizar información de la Catedral de La Plata a medida que los usuarios la recorren físicamente.

Por último, se dieron guías con el objetivo de que la construcción de los modelos sea más sencilla. Estas guías son el producto de la experiencia recabada con el desarrollo del caso de uso.

8.2. Trabajo futuro

Aunque el presente trabajo resuelva el hecho de que las ubicaciones puedan ser fácilmente escalables e integrables motiva el abordaje de otro tipo de problemáticas más complejas que serán realizadas en un trabajo futuro:

- Ubicaciones en otros ámbitos: Cuando hablamos de ubicaciones siempre nos referimos a las mismas desde el punto de vista físico. Sin embargo el término ubicación es utilizado con otras connotaciones en áreas como la hipermedia, la hipermedia física y los ambientes digitales. Esto motiva la idea de extender el concepto de ubicación a diferentes ambientes. Por ejemplo, en el caso de la hipermedia física, un usuario puede estar ubicado en una única ubicación física y en una ubicación en el espacio de hipermedia. Cada ubicación pertenece a ámbitos diferentes y por lo tanto puede variar independientemente de la otra. Por ejemplo, el usuario puede estar posicionado en un mismo lugar físico e irse moviendo por el espacio de hipermedia. También puede ocurrir el caso inverso en el cual el usuario se encuentra moviéndose físicamente y el nodo de hipermedia sobre el cual se encuentra parado no ha cambiado. Para abordar este tipo de problemáticas será necesario modelar el ambiente donde la ubicación se encuentra definida.

- Ubicaciones probabilísticas: La utilización que hemos hecho de las ubicaciones nos permite afirmar el ámbito donde se encuentra un objeto con exactitud. Muchas veces resultaría interesante conocer (en base a alguna información proveniente de los sensores) con que exactitud es que la ubicación modelada representa la ubicación real. En nuestro caso, asumimos que la capa de sensing era la que determinaba la exactitud de la información y por ello perdíamos información probabilística respecto a las ubicaciones. La idea de este aspecto es introducir este tipo de información a las ubicaciones con el fin de que dicha información sea utilizada a la hora de computar operaciones y obtener información respecto a las ubicaciones de los objetos ubicables.

- Herramientas para la construcción de mapas: La construcción de los mapas es una tarea que es realizada en forma manual por el programador.

Esta tarea es tediosa y poco intuitiva debido a que no posee un feedback de como se esta construyendo el mapa e imposible de realizar para una persona sin conocimientos en programación. Por dicho motivo se espera desarrollar un conjunto de editores (utilizando HotDraw [2]) que permitan construir mapas, definiendo los objetos ubicables en cuestión, los sistemas de representación y sus ubicaciones.

Bibliografía

[1] D. Baumer, D. Riehle, W. Siberski, and M. Wulf. The role object pattern.

[2] John Brant and Ralph E. Johnson. Creating tools in hotdraw by composition. In Boris Magnusson, Bertrand Meyer, Jean-Marc Nerson, and Jean-François Perrot, editors, *TOOLS (13)*, pages 445–454. Prentice Hall, 1994.

[3] N. Bulusu, D. Estrin, and J. Heidemann. Tradeoffs in location support systems: The case for quality-expressive location models for applications, 2001.

[4] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, August 1996.

[5] Inc Cincom Systems. Visualworks. http://www.cincomsmalltalk.com/.

[6] Inc Cincom Systems. Taming name spaces, 2000, 2001.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford. *Dijkstra's algorithm*, chapter 24.3, pages 595–601. MIT Press and McGraw-Hill, second edition, 2001.

[8] Anind Kumar Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, 2000. Director-Gregory D. Abowd.

[9] Haibo Hu Dik-Lun. Semantic location modeling for location navigation in mobile environment.

[10] Paul Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004.

[11] Real Academia Española. Diccionario de la lengua española, vigésima segunda edición.

[12] ESRI. Esri shapefile technical description.

[13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley Professional, January 1995.

[14] S. Gessler and K. Jesse. Advanced location modeling to enable sophisticated lbs provisioning in 3g networks, 2001.

[15] Julian Grigera, Andres Fortier, Gustavo Rossi, and Silvia Gordillo. A modular architecture for context sensing. *ainaw*, 2:147–152, 2007.

[16] Kaj Gronbaek, Jannie F. Kristensen, Peter Orbaek, and Mette Agger Eriksen. "physical hypermedia": organising collections of mixed physical and digital material. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 10–19, New York, NY, USA, 2003. ACM Press.

[17] Frank Allan Hansen, Niels Olof Bouvin, Bent G. Christensen, Kaj Gronbaek, Torben Bach Pedersen, and Jevgenij Gagach. Integrating the web and the world: contextual trails on the move. In *HYPERTEXT '04: Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, pages 98–107, New York, NY, USA, 2004. ACM Press.

[18] Simon Harper, Carole A. Goble, and Stephen Pettitt. proximity: Walking the link. *J. Digit. Inf.*, 5(1), 2004.

[19] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001.

[20] Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello. The location stack: A layered model for location in ubiquitous computing. In *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 22, Washington, DC, USA, 2002. IEEE Computer Society.

[21] R. Johnsonand and B. Woolf. The type object pattern, 1997.

[22] J. Kahn, R. Katz, and K. Pister. Emerging challenges: Mobile networking for 'smart dust, 2000.

[23] Max Van Kleek, Kai Kunze, Kurt Partridge, and James Bo Begole. Opf: A distributed context-sensing framework for ubiquitous computing environments. In Hee Yong Youn, Minkoo Kim, and Hiroyuki Morikawa, editors, *UCS*, volume 4239 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2006.

[24] Korf. Artificial intelligence search algorithms. In *Algorithms and Theory of Computation Handbook, CRC Press, 1999*. 1999.

[25] U. Leonhardt. Supporting location-awareness in open distributed systems, 1998.

[26] Ajith K. Narayanan. Realms and states: a framework for location aware mobile computing. In *WMC '01: Proceedings of the 1st international workshop on Mobile commerce*, pages 48–54, New York, NY, USA, 2001. ACM Press.

[27] Daniela Nicklas, Matthias Großmann, Thomas Schwarz, Steffen Volz, and Bernhard Mitschang. A model-based, open architecture for mobile, spatially aware applications. In *SSTD '01: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, pages 117–135, London, UK, 2001. Springer-Verlag.

[28] Daniela Nicklas and Bernhard Mitschang. The nexus augmented world model: An extensible approach for mobile, spatially-aware applications. in: Wang, yingxu (ed.); patel, shushma (ed.); johnston, ronald (ed.): Proceedings of the 7th international conference on object-oriented information systems : OOIS '01 ; calgary, canada, august 27-29, 2001. Technical report, 2001.

[29] James Nord, Kåre Synnes, and Peter Parnes. An Architecture for Location Aware Applications. In *Proceedings of HICSS-35, Big Island, Hawai´i, USA*, january 2002. Nominated to Best Paper.

[30] Lia Oubina. Estructuras algebraicas.

[31] Thomas Pederson. Object location modeling in office environments - first steps.

[32] Stephen Peters and Howard E. Shrobe. Using semantic networks for knowledge representation in an intelligent environment. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, page 323, Washington, DC, USA, 2003. IEEE Computer Society.

[33] Salil Pradhan. Semantic location. *Personal Ubiquitous Comput.*, 4(4):213–216, 2000.

[34] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43, New York, NY, USA, 2000. ACM Press.

[35] Kay Römer. The lighthouse location system for smart dust. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 15–30, New York, NY, USA, 2003. ACM Press.

[36] Bill Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994.

[37] Mike Spreitzer and Marvin Theimer. Providing location information in a ubiquitous computing environment (panel session). In *SOSP '93: Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 270–283, New York, NY, USA, 1993. ACM Press.

[38] Andrew Capella Valerie Bennett. Location based services. wherever you are, wherever you go, get the information you want to know.

[39] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

[40] Wikipedia. Location - wikipedia. http://en.wikipedia.org/wiki/location.

[41] Wikipedia. Semantic network. http://en.wikipedia.org/wiki/semantic_network.

[42] B. Woolf. The null object pattern, 1996.

[43] H. Zimmermann. Osi reference model—the iso model of architecture for open systems interconnection. pages 2–9, 1988.