# Aplicación móvil de sistemas de localización automática de vehículos sensible al contexto


## Alumnos

María Carolina Miceli

A.C. Julián Alberto Rossi

## Director de Tesis

Dra. Silvia Gordillo



PRESENTADO EN CUMPLIMIENTO DE LOS
REQUERIMIENTOS PARA EL GRADO DE
LICENCIATURA EN INFORMÁTICA
DE LA
UNIVERSIDAD NACIONAL DE LA PLATA
LA PLATA, ARGENTINA
MARZO DE 2005

# Agradecimientos

En especial queremos agradecer a nuestros padres que nos dieron la posibilidad de estudiar, nos brindaron un fuerte apoyo y nos animaron en momentos difíciles a lo largo de nuestra vida.

Tampoco queremos olvidarnos de nuestros amigos quienes nos prestaron su atención, preocupación y paciencia durante nuestra carrera.

Por último, al LIFIA y sus miembros quienes nos acompañaron en nuestros años de investigación, compartieron sus experiencias y conocimientos, permitiéndonos dar nuestros primeros pasos en la investigación.

# Contenido

# Índice de figuras

# 1 Introducción

La evolución actual de las tecnologías permite imaginar que en un futuro cercano servicios tales como sistemas de guía, serán parte indispensable de los vehículos. A su vez, se han hecho esfuerzos considerables en el desarrollo de tecnologías móviles y, las comunicaciones inalámbricas dan al usuario la posibilidad de acceder a cualquier clase de información desde cualquier lugar en cualquier momento, lo cual permite usar en aplicaciones móviles procesamiento de datos locales, acceso y visualización de datos.

Los avances en las tecnologías de comunicación y computadoras personales han permitido que actualmente el área de "mobile computing" juegue un importante rol en la vida diaria. Por otro lado, la disponibilidad de tecnologías de sensores ha permitido a las aplicaciones monitorear y usar información del entorno actual, o contexto, recopilado desde sus ambientes, por ejemplo, ayudar a ubicar los autos monitoreados por una agencia de seguros usando información de ubicación capturada con un dispositivo GPS [Trimble] (tecnología que será posteriormente desarrollada).

Las aplicaciones interactivas tradicionales están limitadas a usar solo la información que les es explícitamente suministrada, pero actualmente existe la necesidad de que las aplicaciones utilicen información implícitamente adquirida (o contexto) del entorno actual, y modifiquen su comportamiento en base a esta. Dichas aplicaciones que se adaptan a la información de su contexto son conocidas como "Aplicaciones sensibles al contexto". Se puede requerir que los usuarios expresen toda la información relevante a una situación dada. No obstante, el objetivo de las aplicaciones sensibles al contexto, o aplicaciones que usan el contexto, debería hacer la interacción con las computadoras más fácil. Forzar concientemente a los usuarios a incrementar la cantidad de información que deben ingresar hace esta interacción más difícil y tediosa. Además, es probable que la mayoría de los usuarios no conozcan cual información es potencialmente relevante y, por lo tanto, no conozcan que información proporcionar. Por este motivo se busca facilitar a los usuarios la interacción con las computadoras y el entorno, recolectando información contextual implícitamente de manera automatizada [Dey, 00].

La necesidad de relevar información de contexto es aún mayor cuando nos movemos. La computación móvil da a los usuarios la posibilidad de acceder a un conjunto de información y servicios que requieran, cuando y donde lo deseen. Cuando las computadoras son utilizadas en tal amplia variedad de situaciones aparecen nuevos problemas y la necesidad de representar la información de contexto es clara: usuarios tratando diferente información desde los mismos servicios en diferentes situaciones. El contexto puede ser usado para determinar que información o servicios hacer disponibles o brindar a los usuarios en una situación determinada. De esta manera, los usuarios pueden moverse a través del entorno mientras llevan su poder de cómputo con ellos. La combinación de estas características con el uso de comunicaciones inalámbricas permite a los usuarios acceder a información y servicios no disponibles directamente en sus dispositivos de cálculo portátiles. La movilidad hace que el entorno del usuario, tal como su ubicación y las personas que lo rodean, sea más dinámico.

En esta tesis, se estudiará la problemática de las aplicaciones móviles aplicándolas en un caso particular, el de un sistema de localización automática de vehículos (AVL) que permite monitorear la posición exacta de vehículos en movimiento

sobre mapas digitales, explorando los problemas pertinentes relacionados con el desarrollo de aplicaciones sensibles al contexto diseñándola para el uso en entornos dinámicos, donde los vehículos participantes se desplazan por dicho mapa.

Este sistema de localización constará de una central, la cual debe conocer la posición de cada vehículo del sistema; a su vez, los vehículos son de dos clases: monitoreados y los patrulleros.

Los vehículos monitoreados son aquellos que contratan un servicio de seguimiento a una empresa para que en caso de robo pueda ser localizado y recuperado. Dichos vehículos envían señales de su posición a la central.

Los patrulleros son vehículos pertenecientes a la empresa que recorren la ciudad vigilando los vehículos monitoreados y deben enviar señales de su posición a la central. Estos contarán con algún dispositivo que les permitirá manejar y visualizar cierta cantidad de información sobre el resto de los vehículos y datos de la ciudad, ruta, etc., en la que estos se encuentran, siendo esta adaptada a su entorno.

El sistema se definirá abstrayendo los diferentes subsistemas que lo componen y las tecnologías necesarias de manera de obtener una simulación basada en las decisiones de diseño. El prototipo será descompuesto en servicios independientes obteniendo como resultado módulos pequeños.

Las comunicaciones inalámbricas son otra característica clave para el sistema dado que permite la conexión con fuentes de datos distantes. Es por esto que serán investigadas las tecnologías inalámbricas existentes.

## 1.1 Resumen de los capítulos

En el capítulo 2 se presentan los sistemas AVL tradicionales, sus funcionalidades y necesidades para su funcionamiento. Para concluir se hace un relevamiento del sistema de posicionamiento más utilizado en sistemas AVL (GPS).

El capítulo 3 describe los Sistemas de información Geográfica (GIS), sus operaciones habituales, los modelos más utilizados y la conveniencia de su utilización en sistemas móviles donde sea de relevancia la ubicación, como ocurre en este caso.

En el capítulo 4 se presentan las actuales definiciones de sistemas móviles y sensibilidad al contexto. En primer lugar se establece una discusión sobre lo que es Mobile computing según diferentes autores, se mencionan los problemas extras asociados al trabajar con sistemas móviles y se describen los sistemas de ubicación móvil, que son sistemas móviles que tienen en cuenta la ubicación. Luego se mencionan los actuales dispositivos móviles, las tecnologías de sensores existentes y las diferentes tecnologías de comunicaciones. Para finalizar el capítulo se exponen las definiciones de contexto según diferentes autores y las diferentes posibles propiedades a tener en cuenta en ellos, para posteriormente describir los sistemas sensibles al contexto.

El capítulo 5 describe el sistema de AVL a desarrollar, se definen las diferentes partes intervinientes, funcionalidades y servicios existentes en cada una de ellas. Con respecto a lo que es la aplicación móvil sensible al contexto, además de describir sus funcionalidades, se muestran las adaptaciones que sufrirán cada una de ellas en base a los contextos tenidos en cuenta.

El capítulo 6 comienza con la descripción de un modelo de base para un sistema AVL. Luego, se hace una breve descripción de los modelos existentes en el modelado

de contexto, para posteriormente tomar los puntos favorables de ellos y hacer la elección propia del modelo a utilizar. Para finalizar, el capítulo describe los diferentes módulos del sistema, tanto la central receptora como los patrulleros y la comunicación que existirá entre ellos.

En el capítulo 7 se muestra la implementación realizada tanto de la central como los usuarios móviles, se describen las tecnologías utilizadas y los problemas encontrados al hacer su realización.

Por último, el capítulo 8 presenta las conclusiones y trabajos futuros.

De manera adicional, el Apéndice I releva una de las tecnologías utilizadas en le prototipo móvil: J2ME y el Apéndice II describe los elementos de la suite de componentes Map-Object que fue el producto GIS utilizado en el desarrollo.

# 2 Sistemas AVL (Localización automática de vehículos)

## 2.1 Introducción

En esta sección se explicarán brevemente las características de los sistemas de localización automática de vehículos y algunas de las tecnologías necesarias para su implantación.

## 2.2 Introducción a AVL

Un sistema de localización automática de vehículos permite saber la posición exacta de vehículos en movimiento sobre mapas digitales, por medio de información satelital, usando tecnología de sensores de posicionamiento. La tecnología de posicionamiento más ampliamente usada es el GPS (Sistema de posicionamiento global), la cual será desarrollada en la sección 2.3.

Los vehículos cuentan con una unidad GPS y un dispositivo de comunicación. Con el sensor de posicionamiento toman su posición exacta en un instante de tiempo y transmiten esta posición por el dispositivo de comunicación. El envío de la posición se hace por medio de transmisores de radio convencionales o troncales, teléfonos celulares, ó usando algún satélite comercial (distinto a los satélites GPS que son emisores únicamente). La Figura 2.1 muestra un ejemplo de captura y envío de posiciones.

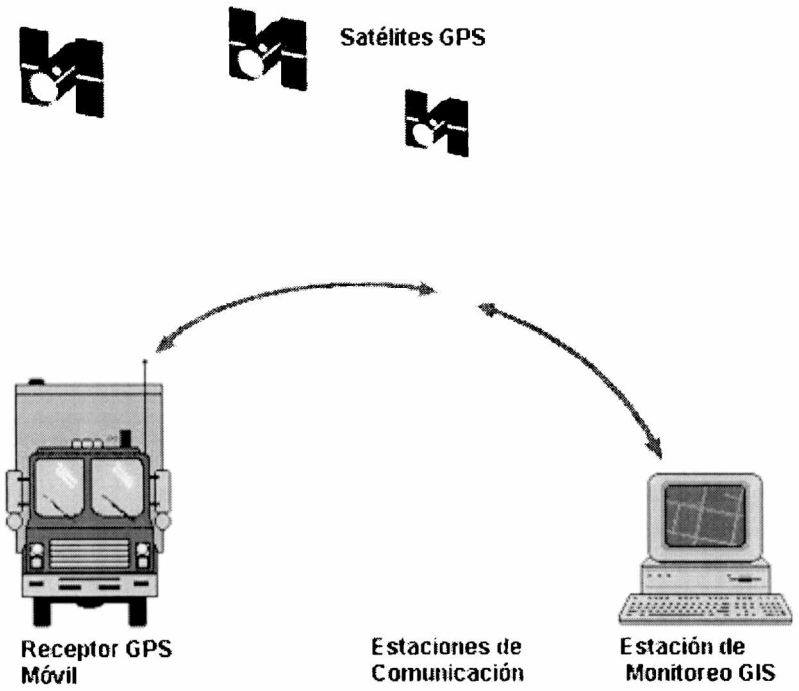


**Figura 2.1: Localización y envío de señales**

Una central receptora recibe dicha información y la utiliza de diferentes maneras dependiendo del servicio provisto al vehículo o la situación en la que este se encuentre.

Algunas de las operaciones existentes en sistemas AVL, son:

- Conocer el lugar donde se encuentra cada vehículo.
- Medir la distancia entre un vehículo y un punto determinado.
- Calcular tiempos de llegada.
- Ubicar puntos de interés.
- Visualizar en una PC todo el recorrido de los vehículos.
- Informar la trayectoria de un vehículo, lugar y duración de paradas.
- Detectar el estado de un vehículo (robado o no).
- Almacenar información para análisis posteriores.

## *2.3 Sistema GPS*

**El siguiente es un relevamiento hecho en base a la información provista por Trimble Navigation Limited [Trimble].**

GPS se refiere a las iniciales de Global Positioning System (Posicionamiento global Satelital), consiste en 24 satélites militares norteamericanos (más 3 de repuesto) que envían pulsos a cierta frecuencia para que unidades en tierra determinen su posición en cualquier parte del mundo durante las 24 horas en cualquier condición climática, y que después de la guerra fría se usan en aplicaciones.

### 2.3.1 Triangulación

Aunque pueda parecer improbable, la idea general detrás del GPS es utilizar los satélites en el espacio como puntos de referencia para ubicaciones aquí en la tierra.

Esto se logra mediante una medición muy exacta de nuestra distancia hacia al menos tres satélites, lo que nos permite "**triangular**" nuestra posición en cualquier parte de la tierra.

**Por ejemplo:**

Suponga que mide la distancia a un primer satélite y resulta ser de 11.000 millas (20.000 Km).

Si se encuentra a 11.000 millas de un satélite determinado, no puede estar en cualquier punto del universo sino que esto limita su posición a la superficie de una esfera que tiene como centro dicho satélite y cuyo radio es de 11.000 millas, como muestra la Figura 2.2.

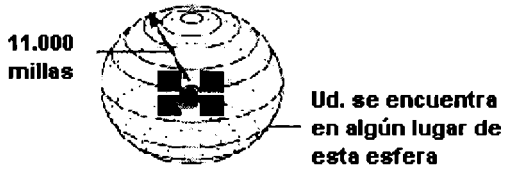


**Figura 2.2: Esfera con centro en el satélite.**

Luego mide la distancia a un segundo satélite y descubre que se encuentra a 12.000 millas del mismo.

Esto determina que no se encuentra solamente en la primera esfera, correspondiente al primer satélite, sino también sobre otra esfera ubicada a 12.000 millas del segundo satélite. En otras palabras, esta en algún lugar de la circunferencia que resulta de la intersección de las dos esferas. La Figura 2.3 muestra dicho resultado.

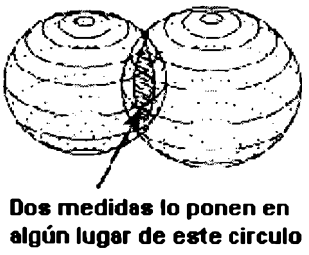


**Figura 2.3: Intersección entre esferas con centro en dos satélites.**

Si ahora mide la distancia a un tercer satélite y descubre que esta a 13.000 millas del mismo, esto limita su posición aún más, a los dos puntos en los cuales la esfera de 13.000 millas corta la circunferencia que resulta de la intersección de las dos primeras esferas, como muestra la Figura 2.4.


**Figura 2.4: Puntos resultantes de la intersección entre tres esferas.**

O sea, que midiendo la distancia a tres satélites se limita el posicionamiento a solo dos puntos posibles.

Para decidir cual de ellos es la posición real, se podría efectuar una nueva medición a un cuarto satélite. Pero normalmente uno de los dos puntos posibles resulta ser muy improbable por su ubicación demasiado lejana de la superficie terrestre y puede ser descartado sin necesidad de mediciones posteriores. Sin embargo una cuarta medición es muy conveniente por una razón analizada posteriormente.

Se sabe que la posición se calcula a partir de la medición de la distancia hasta por lo menos tres satélites. Pero, para medir dicha distancia hacia algo que esta flotando en el espacio se debe calcular el tiempo que tarda una señal emitida por el satélite en llegar hasta el receptor de GPS.

La idea es similar a un problema simple de velocidad tal como "Si un auto viaja a 60 kilómetros por hora durante dos horas, ¿qué distancia recorrió?

**Velocidad (60 km/h) x Tiempo (2 horas) = Distancia (120 km)**

En el caso del GPS se esta midiendo una señal de radio, que viaja a la velocidad de la luz, alrededor de 300.000 km por segundo. El problema es medir el tiempo de viaje de la señal, la cual viene demasiado rápido.

### 2.3.2 El problema del tiempo

El problema de la medición de ese tiempo es algo complicado dado que los tiempos son extremadamente cortos. Si el satélite estuviera a unos 20.000 km de altura, el tiempo total de viaje de la señal hacia el receptor sería de algo más de 0.06 segundos. Por lo cual se necesitan relojes muy precisos.

Suponga que tanto el GPS como el satélite, generan una señal auditiva en el mismo instante, y que parados al lado del receptor de GPS se pueden oír ambas señales (Obviamente es imposible "oír" esas señales porque el sonido no se propaga en el vacío).

Se oirían dos versiones de la señal. Una de ellas inmediatamente, generada por el receptor GPS y la otra con cierto atraso, proveniente del satélite, porque tuvo que recorrer alrededor de 20.000 km para llegar. Se puede decir que ambas señales no están sincronizadas.

Si se quiere saber cual es la magnitud de la demora de la señal proveniente del satélite se puede retardar la emisión de la señal del GPS hasta lograr la perfecta sincronización con la señal que viene del satélite.

El tiempo de retardo necesario para sincronizar ambas señales es igual al tiempo de viaje de la señal proveniente del satélite. Si esta es de 0.06 segundos. Conociendo este tiempo, es multiplicado por la velocidad de la luz y se obtiene la distancia hasta el satélite.

**Tiempo de retardo (0.06 seg) x Vel. de la luz (300.000 km/seg) = Dist. (18.000 km)**

Básicamente esta es la forma en la que funciona un GPS.

### 2.3.3 El código de transmisión

La señal emitida tanto por un GPS como por el satélite es algo llamado "Código Pseudo Aleatorio1" (Pseudo Random Code).

Este Código Pseudo Aleatorio es una parte fundamental del GPS. Físicamente solo se trata de una secuencia o código digital muy complicado. O sea una señal que contiene una sucesión complicada de pulsos "on" y "off" (unos y ceros).

Hay varias y muy buenas razones para tal complejidad. La complejidad del código ayuda a asegurar que el receptor de GPS no se sintonice accidentalmente con alguna otra señal. Siendo el modelo tan complejo es altamente improbable que una señal cualquiera pueda tener exactamente la misma secuencia.

Dado que cada uno de los satélites tiene su propio y único Código Pseudo Aleatorio, esta complejidad también garantiza que el receptor no se confunda accidentalmente de satélite. De esa manera, también es posible que todos los satélites trasmitan en la misma frecuencia sin interferirse mutuamente. Esto también complica a cualquiera que intente interferir el sistema desde el exterior al mismo.

---

[1] Aleatorio: algo generado al azar.

Hasta el momento se asumió que el GPS y el satélite comienzan la emisión de la señal exactamente al mismo tiempo. Si la medición del tiempo de viaje de una señal de radio es clave para el GPS, los relojes que se empleen deben ser exactísimos, dado que si miden con un desvío de un milésimo de segundo, a la velocidad de la luz, ello se traduce en un error de 300 km.

Por el lado de los satélites, el timing es casi perfecto porque llevan a bordo relojes atómicos de increíble precisión.

Recuerde que ambos, el satélite y el receptor GPS, deben ser capaces de sincronizar sus Códigos Pseudo Aleatorios para que el sistema funcione.

Si los receptores GPS tuvieran que alojar relojes atómicos (Cuyo costo está por encima de los 50 a 100.000 U$S) la tecnología resultaría demasiado costosa y nadie podría acceder a ellos. Pero existe una solución a este problema que radica en realizar cuatro mediciones.

Si todo fuera perfecto (es decir que los relojes de los receptores GPS lo fueran), entonces todos los rangos (distancias) a los satélites se intersecarían en un único punto (que indica nuestra posición). Pero con relojes imperfectos, una cuarta medición efectuada como control cruzado, no intersecará con los tres primeros.

De esa manera la computadora del receptor GPS detectará la discrepancia y atribuirá la diferencia a una sincronización imperfecta con la hora universal.

Dado que cualquier discrepancia con la hora universal afectará a las cuatro mediciones, el receptor buscará un factor de corrección único que siendo aplicado a sus mediciones de tiempo hará que los rangos coincidan en un solo punto.

Dicha corrección permitirá al reloj del receptor ajustarse nuevamente a la hora universal.

Una vez que el receptor de GPS aplica dicha corrección al resto de sus mediciones, se obtiene un posicionamiento preciso.

Una consecuencia de este principio es que cualquier GPS decente debe ser capaz de sintonizar al menos cuatro satélites de manera simultánea. En la práctica, casi todos los GPS en venta actualmente, acceden a más de 6, y hasta a 12, satélites simultáneamente.

Ahora bien, con el Código Pseudo Aleatorio como un pulso confiable para asegurar la medición correcta del tiempo de la señal y la medición adicional como elemento de sincronización con la hora universal, se tiene todo lo necesario para medir nuestra distancia a un satélite en el espacio.

Pero, para que la triangulación funcione se necesita conocer no sólo la distancia sino que se debe conocer dónde están los satélites con toda exactitud.

A lo largo de esta descripción se ha asumido que se conoce dónde están los satélites en sus órbitas y de esa manera pueden utilizarse como puntos de referencia.

Todos ellos están flotando a unos 20.000 km de altura en el espacio. Esta altura es en realidad un gran beneficio para este caso, porque algo que está a esa altura está bien despejado de la atmósfera. Eso significa que orbitará de manera regular y predecible mediante ecuaciones matemáticas sencillas.

La Fuerza Aérea de los EEUU colocó cada satélite de GPS en una órbita muy precisa, de acuerdo al Plan Maestro de GPS.

En tierra, todos los receptores de GPS tienen un almanaque programado en sus computadoras que les informan donde está cada satélite en el espacio, en cada momento. La Figura 2.5 muestra una antena receptora que interactúa con un satélite.

**Figura 2.5: Receptor GPS.**

Se utilizan radares muy precisos para controlar constantemente la exacta altura, posición y velocidad de cada satélite.

Los errores que se controlan son los llamados errores de efemérides, o sea evolución orbital de los satélites. Estos errores se generan por influencias gravitacionales del sol y de la luna y por la presión de la radiación solar sobre los satélites. Estos errores son generalmente muy sutiles pero si se quiere una gran exactitud se deben tener en cuenta.

Una vez que el Departamento de Defensa ha medido la posición exacta de un satélite, vuelven a enviar dicha información al propio satélite. De esa manera el satélite incluye su nueva posición corregida en la información que transmite a través de sus señales a los GPS. La Figura 2.6 muestra el envío de información para corregir la posición del satélite.

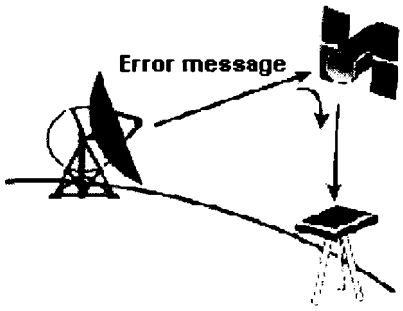


**Figura 2.6: Corrección de posición del satélite.**

Esto significa que la señal que recibe un receptor de GPS no es solamente un Código Pseudo Aleatorio con fines de timing. También contiene un mensaje de navegación con información sobre la órbita exacta del satélite

Con un timing perfecto y la posición exacta del satélite se puede pensar en efectuar cálculos perfectos de posicionamiento. Sin embargo se deben resolver otros problemas.

## 2.3.4 Corrección de errores

Hasta ahora se ha tratado los cálculos del sistema GPS como si todo el proceso ocurriera en el vacío. Pero en el mundo real hay muchas cosas que le pueden suceder a una señal de GPS para transformarla en algo menos que matemáticamente perfecta.

Para aprovechar al máximo las ventajas del sistema un buen receptor de GPS debe tener en cuenta una amplia variedad de errores posibles.

### Viaje a través de la atmósfera

En primer lugar, una de las presunciones básicas que se ha usado no es exactamente cierta. Se ha afirmando que es posible calcular la distancia a un satélite multiplicando el tiempo de viaje de su señal por la velocidad de la luz. Pero la velocidad de la luz sólo es constante en el vacío.

Una señal de GPS pasa a través de partículas cargadas en su paso por la ionosfera y luego al pasar a través de vapor de agua en la troposfera pierde algo de velocidad, creando el mismo efecto que un error de precisión en los relojes. La Figura 7 muestra el viaje de la señal a través de las diferentes capas de la atmósfera.

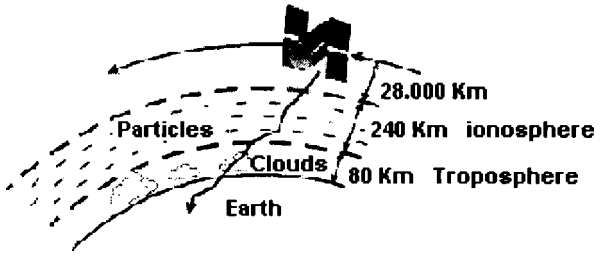


**Figura 2.7: Viaje a través de la atmósfera.**

Existen dos maneras de minimizar este tipo de error. Por un lado, predecir cual sería el error tipo de un día promedio. A esto se lo llama modelación pero, por supuesto, las condiciones atmosféricas raramente se ajustan exactamente el promedio previsto.

Otra forma de manejar los errores inducidos por la atmósfera es comparar la velocidad relativa de dos señales diferentes. Esta medición de doble frecuencia es muy sofisticada y solo es posible en receptores GPS muy avanzados.

### Viaje sobre la tierra

Los problemas para la señal de GPS no terminan cuando llega a la tierra. La señal puede rebotar varias veces debido a obstrucciones locales antes de ser captada por el receptor GPS, como muestra la Figura 2.8.

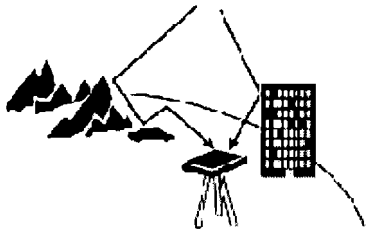

**Figura 2.8: Viaje sobre la tierra.**

Este error es similar al de las señales fantasma que se pueden ver en la recepción de televisión. Los buenos receptores GPS utilizan sofisticados sistemas de rechazo para minimizar este problema.

### Problemas en el satélite

Aún siendo los satélites muy sofisticados no tienen en cuenta minúsculos errores en el sistema.

Los relojes atómicos que utilizan son muy precisos, pero no son perfectos. Pueden ocurrir minúsculas discrepancias que se transforman en errores de medición del tiempo de viaje de las señales.

Y, aunque la posición de los satélites es controlada permanentemente, tampoco pueden ser controlados a cada instante. De esa manera pequeñas variaciones de posición o de efemérides pueden ocurrir entre los tiempos de monitoreo.

### Algunos ángulos son mejores que otros

La geometría básica por si misma puede magnificar estos errores mediante un principio denominado "Dilación Geométrica de la Precisión", o DGDP

En general suele haber más satélites disponibles que los que el receptor GPS necesita para fijar una posición, de manera que el receptor toma algunos e ignora al resto.

Si el receptor toma satélites que están muy juntos en el espacio, las circunferencias de intersección que definen la posición se cruzarán a ángulos con muy escasa diferencia entre sí. Esto incrementa el área gris o margen de error acerca de una posición. La Figura 9 muestra el resultado de la intersección de satélites cercanos.

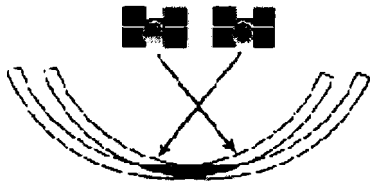


**Figura 2.9: Intersección de satélites cercanos.**

Si el receptor toma satélites que están ampliamente separados, las circunferencias intersecan a ángulos prácticamente rectos y ello minimiza el margen de error. La Figura 2.10 muestra tal resultado.


**Figura 2.10: Intersección de satélites distantes**

Los buenos receptores son capaces de determinar cuales son los satélites que dan el menor error por Dilución Geométrica de la Precisión.

## Errores Intencionales

Aunque resulte difícil de creer, el mismo Gobierno que pudo gastar 12.000 Millones de dólares para desarrollar el sistema de navegación más exacto del mundo, está degradando intencionalmente su exactitud. Dicha política se denomina "Disponibilidad Selectiva" y pretende asegurar que ninguna fuerza hostil o grupo terrorista pueda utilizar el GPS para fabricar armas certeras.

Básicamente, se introduce cierto "ruido" en los datos del reloj satelital, lo que a su vez se traduce en errores en los cálculos de posición.

Estos errores en su conjunto son la mayor fuente unitaria de error del sistema GPS. Los receptores de uso militar utilizan una clave encriptada para eliminar la Disponibilidad Selectiva y son, por ello, mucho más exactos.

Afortunadamente todos esos errores no suman demasiado error total. Existe una forma de GPS, denominada GPS Diferencial, que reduce significativamente estos problemas.

# 3 Sistemas de Información Geográfica (GIS)

## 3.1 Introducción

El área de Mobile computing(Computación móvil) se encuentra relacionada con sistemas de información geográfica a través de sus servicios de ubicación móvil. Las tecnologías de análisis espacial desarrolladas en GIS han sido reformuladas por la velocidad y escalabilidad requerida por los servicios de ubicación móvil.

En el presente capítulo se describirán las tecnologías de información geográfica y como se encuentran vinculadas en el desarrollo de aplicaciones móviles, y en particular en la solución de un sistema de seguimiento vehicular.

## 3.2 Sistemas de Información Geográfica: Definición

Según Aronoff [Aronoff, 89] GIS es un sistema basado en computadoras que proporciona 4 capacidades para manejar datos geo-referenciados, entrada; administración de datos (almacenamiento y recuperación de datos); manipulación y análisis; y salida.

Una clasificación de los elementos básicos del estudio científico que puede ser investigada usando GIS son los mencionados por Rhind [Rhind, 90]. Estos son:

1) Ubicación. ¿Qué hay en...?
2) Condición. ¿Dónde se encuentra?
3) Tendencia. ¿Qué ha cambiado desde...?
4) Caminos. ¿Cuál es el mejor camino?
5) Patrones. ¿Qué patrones de distribución espacial existen?
6) Modelado. ¿Qué sucede si...?

Las preguntas de ubicación involucran consultas a bases de datos para determinar los tipos de características que ocurren en un lugar dado.

Las preguntas de condición involucran la búsqueda de sitios que tiene ciertas características. El cambio en el tiempo es monitoreado usando las preguntas de tendencia.

Las últimas 3 clases involucran análisis espaciales complejos. La mayoría de las búsquedas de caminos de un lugar a otro por algún criterio se resuelven con algoritmos de búsqueda de caminos. La descripción y comparación de diferentes fenómenos y los procesos con los cuales cuenta para esa distribución pueden ser hecho usando preguntas de patrones. Las preguntas de modelado nos permiten experimentar con diferentes escenarios.

Desde un punto de vista práctico un Sistema de Información Geográfica es un sistema informático capaz de realizar una gestión completa de datos geográficos referenciados. Por referenciados se entiende que estos datos geográficos o mapas tienen coordenadas geográficas reales asociadas, las cuales nos permiten manejar y hacer análisis con datos reales como longitudes, perímetros o áreas. Todos estos datos alfanuméricos asociados a los mapas más los que queramos añadirle los gestiona una base de datos integrada con el GIS.

## 3.3 Funcionalidad del GIS:

En grandes líneas se puede decir que un GIS permite:

- Construir datos geográficos: Mediante datos geométricos existentes en CAD, o capturándolos por digitalización, vectorización de imágenes, GPS, etc., el sistema permite depurarlos y estructurarlos topológicamente, asociándolos con bases de datos alfanuméricas. De esta forma se obtienen datos espaciales listos para su uso en el análisis.

- Modelado cartográfico: Creación de nuevos mapas a partir de mapas existentes, combinando atributos del terreno como pendiente, vegetación, tipo de suelo, etc. Mediante un modelo matemático se pueden crear nuevas variables, como un índice de erosionabilidad, de riesgo de incendios, etc.

- Analizar los mapas estructurados en combinación con bases de datos asociadas.: Consultar las bases de datos asociadas permite seleccionar datos de interés, ver resultados interactivamente eligiendo la simbología en función de los atributos asociados y producir cartografía de calidad.

## 3.4 Tipos de datos geográficos

Los tipos de datos geográficos son la base para representar los fenómenos geográficos del mundo real. Existen dos modelos para representar datos geográficos digitales en computadoras: vectorial y raster.

### 3.4.1 Modelo Vectorial

En el modelo de datos vectorial, los datos geográficos son almacenados como puntos, líneas y polígonos cuyas representaciones están basadas en coordenadas x e y. Cada objeto geográfico tiene asociado información que lo describe. Por ejemplo, un camino tiene asociado su nombre, velocidad máxima, etc.

#### 3.4.1.1 Puntos y nodos

Los puntos se refieren a objetos que tienen ubicación y atributos de información, pero no son lo suficientemente grandes para ser representados como áreas. Dependiendo de la escala del mapa un mismo objeto puede ser representado como un punto o un polígono. Por ejemplo una ciudad puede ser representada como un punto en un mapa a gran escala (a nivel país) pero como un polígono si esta escala es menor (a nivel regional). La Figura 3.1 muestra un ejemplo de representación de un punto.

Los nodos son un tipo especial de punto que representa una unión entre líneas o el punto final de una línea.

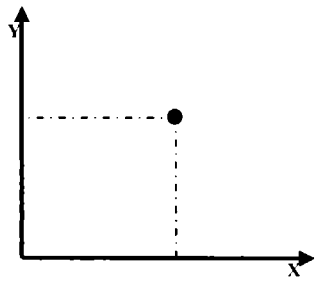


**Figura 3.1: Punto**

### 3.4.1.2 Líneas y arcos

Las líneas son objetos que no tienen área, que comienzan y finalizan con un nodo. Un arco es un segmento que une dos nodos. La Figura 3.2 gráfica un ejemplo de una línea con tres arcos.

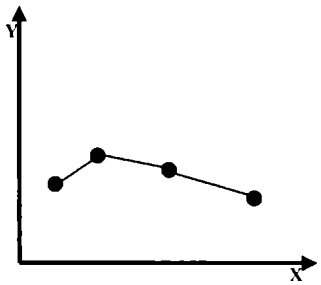


**Figura 3.2: Línea y arcos**

### 3.4.1.3 Polígonos

Un polígono es un conjunto de líneas conectadas que forman una figura matemática cerrada. Los polígonos pueden tener cualquier número de puntos, forma y tamaño. A su vez, pueden tener huecos, contener otros polígonos, y ser directamente adyacentes a otros polígonos. La Figura 3.3 muestra un polígono simple, sin huecos.

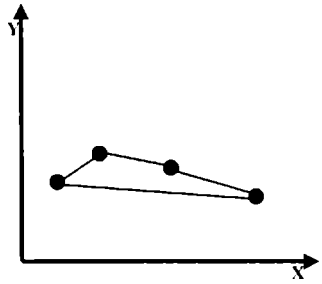


**Figura 3.3: Polígono**

## 3.4.2 Modelo Raster

En el modelo de datos raster, el espacio geográfico es divido en una grilla de celdas normalmente cuadradas (a veces rectangulares). Toda variación geográfica es expresada asignando propiedades o atributos a esas células. Las células también son

llamadas píxel. Dependiendo del nivel de detalle para la visualización, el tamaño de las células varía. Un mapa raster muy simple es mostrado en la Figura 3.4.

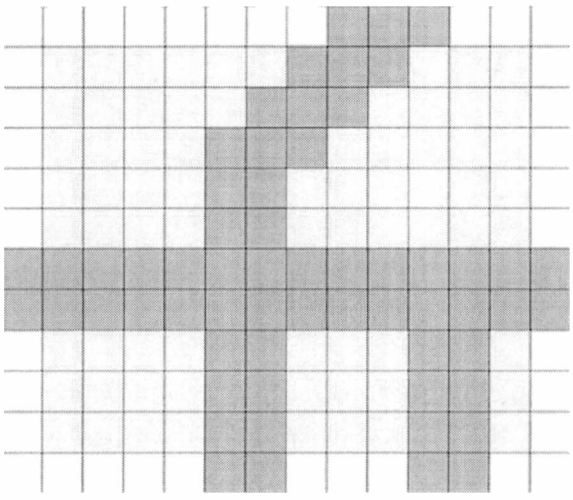

**Figura 3.4: Mapa raster simple**

## 3.4.3 Ventajas y desventajas de los modelos

### 3.4.3.1 Modelo vectorial

**Ventajas**

*La representación de la realidad es buena*

Los mapas vectoriales son mejores para representar la forma precisa de características discretas tal como caminos.

*Las estructuras de datos son compactas*

Los datos en el modelo vectorial ocupan menos espacio de almacenamiento, lo cual los hace mejores si se quiere hacer trasferencias, a través de la red.

*Amplia gama de operaciones espaciales*

Debido a la aproximación matemática del modelo vectorial, soporta muchas operaciones y tiene mayor exactitud que el modelo raster.

**Desventajas**

*Estructuras de datos complejas*

Como contrapartida de que las estructuras de datos de vector ocupen poco espacio, su estructura de datos es muy compleja.

*Simulación de variaciones continuas dificultosa*

La simulación de variaciones continuas es dificultosa para datos de tipo vector.

*El análisis espacial es dificultoso*

Los mapas vectoriales no son buenos como los mapas raster para análisis geográfico que involucre coincidencia espacial, análisis de superficie, proximidad, o camino de menor costo.

### 3.4.3.2 Modelo raster

**Ventajas**

*Estructura de datos simple*

La estructura de datos raster es muy simple dado que consiste de píxeles en una grilla. Por su simplicidad es muy utilizada por productos comerciales.

*Fácil actualización*

La actualización de mapas es muy fácil con mapas raster. Una actualización es hecha por píxel.

*Clases de análisis espaciales variados*

Aunque no se encuentran disponibles todas las operaciones de análisis espacial, operaciones simples como áreas y perímetros pueden ser efectuadas más eficientemente, esto se debe a que las operaciones son simplemente un conteo.

*Forma y tamaño uniforme*

Los mapas raster son fáciles de generar porque ellos son una colección de píxeles. Generar un mapa es un proceso de pintar un píxel en la región requerida.

*La tecnología es económica*

Los mapas raster usan tecnología más económica, porque estas solo requieren al cliente un dispositivo para pintar píxeles en una grilla de resolución variada.

**Desventajas**

*Datos que ocupan mucho espacio*

Dependiendo del tamaño y la resolución, los mapas raster pueden tener grandes cantidades de información. Cada píxel puede contener atributos, y en un mapa de alta resolución hay millones de píxeles.

*Visualización poco atractiva*

La visualización en mapas vectoriales puede ser poco atractiva para el ojo humano ya que muchas veces los píxel (si no son de un tamaño muy reducido) son percibidos.

*Complicada transformación de proyecciones*

Para cambiar las proyecciones a gráficos raster, se requiere que el servidor genere un nuevo mapa, esto se debe a que la transformación es demasiado complicada.

### 3.4.4 Conclusión

En servicios de ubicación móvil el ancho de banda es limitado, con lo cual la transferencia de datos raster no es deseable, ya que estos ocupan gran cantidad de espacio.

Dado que el análisis de mapas vectoriales es mejor en lo que se refiere a servicios de ubicación móvil, es deseable que lo productos que se usan soporten este tipo de estructura y usen datos raster solo para visualización.

## *3.5 Capas de información geográfica (Layers)*

Cuando se modela el mundo real para representaciones de GIS es conveniente agrupar entidades del mismo tipo juntas. Una colección de entidades del mismo tipo es llamada layer, clase o capa indistintamente. Agrupar entidades del mismo tipo hace el almacenamiento de tipos de datos geográficos más eficiente, también facilita la implementación de reglas para validación de los datos y la construcción de relaciones entre entidades. Según Longley [Longley, 02] un layer es una colección de entidades geográficas del mismo tipo geométrico (por ejemplo líneas, puntos, o polígonos), que están lógicamente relacionadas. Que las entidades estén lógicamente relacionadas significa que tratan información de un mismo tema, por ejemplo los ríos pueden ser presentados en un layer ya que representan una capa temática. Para poder representar un layer con más de un tipo geométrico existen los layers compuestos, los cuales pueden combinar layers de diferentes tipos geométricos.

Tanto el modelo de datos raster como el modelo de datos de vector usan layers para manejar entidades geográficas

## *3.6 Análisis espacial*

### 3.6.1 Definición

Es el proceso por el cual se transforman los datos en información útil. El análisis espacial incluye todas las transformaciones, manipulaciones, y métodos que pueden ser aplicados a datos geográficos para agregarles valor, soporte de decisiones, y revelar patrones y anomalías que no son obvios [Longley, 02].

### 3.6.2 Tipos de análisis espaciales

#### 3.6.2.1 Consultas

Las consultas son las operaciones más básicas del análisis espacial, en las cuales GIS es usado para retornar requerimientos simples hechos por el usuario. No se producen cambios en la base de datos y ningún nuevo dato es ingresado. Ejemplos de estos son, cantidad de casas a un 1 km. de un punto específico, o cual es la ciudad más cercana a un punto.

### 3.6.2.2 Mediciones

Las mediciones son simplemente valores numéricos que describen aspectos de datos geográficos. Incluyen medidas de propiedades simples de objetos, como por ejemplo largo o área, y relaciones entre pares de objetos.

### 3.6.2.3 Transformaciones

Las transformaciones son métodos simples de análisis espacial que cambian los datos, combinándolos o comparándolos para obtener nuevos datos. Las transformaciones usan reglas lógicas, aritméticas o geométricas, e incluyen operaciones para convertir datos raster en datos vector, o viceversa. Un ejemplo es estimar las precipitaciones, temperatura u otro atributo en lugares donde no hay pronóstico del tiempo y no existen medidas directas de estas variables

### 3.6.2.4 Síntesis descriptivas

Las síntesis descriptivas intentan capturar la esencia de un conjunto de datos en uno o dos valores. Son equivalentes a la desviación estándar y la media en el análisis estadístico. Un ejemplo sería en una zona rural ubicar el centro geográfico para ubicar allí el pueblo. Donde el centro geográfico no se obtiene simplemente como centro sino que se calcula teniendo en cuenta la cantidad de habitantes en cada zona, de forma que la mayor cantidad sea beneficiada.

### 3.6.2.5 Técnicas de optimización

Las técnicas de optimización fueron diseñadas para seleccionar ubicaciones ideales para objetos dados según cierto criterio definido. Un ejemplo en el que se suele utilizar es cuando se quiere ubicar un negocio, y se hace un estudio para ver en que zona funcionaría bien.

### 3.6.2.6 Test de Hipótesis

Se enfoca en el proceso de razonamiento de resultados de una muestra limitada para hacer generalizaciones sobre una población completa. Esto permite determinar si un patrón dado descubierto en una muestra se cumple en toda la población.

El análisis espacial requiere una base de datos de mapas digitales y un conjunto de herramientas para ejecutar operaciones espaciales sobre los datos. Las operaciones espaciales incluyen algunas de las que serán utilizadas para el área de mobile computing, esas operaciones son geocodificación (el proceso de ver una posición a partir de una dirección), reverse geocoding (el proceso de ver una dirección a partir de una posición), routing (calcular una ruta entre dos posiciones), map rendering (traducir el área de la base de datos de mapas en un vector o raster), y varias otras.

## 3.7 Sistema de Referencias

La siguiente sección es un relevamiento hecho en base a la información provista por Gordillo et al. [Gordillo, 00].

La posición de un objeto en el mundo real está dada por una medida y un punto u objeto que se toma como referencia. Esta medida carece de todo sentido en forma aislada, es decir, es inútil si no se tiene información adicional para poder interpretarla. Una medida debe ser interpretada dentro de un contexto; a este contexto se lo denomina Sistema de Referencia.

El sistema de referencia define el contexto mediante dos elementos fundamentales: un centro de referencia y un método que establece cómo se relacionan el resto de los puntos con ese centro. Por ejemplo, el sistema de coordenadas esférico define como centro la intersección de los ejes x, y, z; las coordenadas que utiliza consta de tres medidas: una escalar y dos angulares. La medida escalar indica la distancia del punto al centro de referencia; una angular define el ángulo formado por la recta del centro al punto y el eje z, la otra angular especifica el ángulo formado por esa recta y el eje x; con esto está definiendo el método de ubicación de los puntos.

Existen muchos sistemas de referencia, cada uno de los cuales define su centro y el método en base al cual se lo relacionará con los demás puntos. Algunos de ellos son el ortogonal, el cilíndrico, y el esférico. Todos ellos utilizan el mismo centro y lo definen como un punto pero difieren en el método de ubicación de los puntos. La Figura 3.5 muestra un ejemplo de sistema de referencia elíptico.

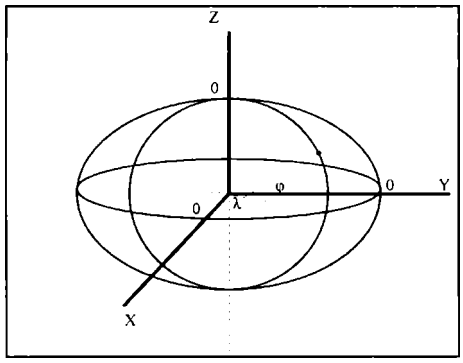


**Figura 3.5: Sistema de Referencia Elíptico**

Varios estudios realizados demostraron que la forma de la tierra no es exactamente un elipsoide, sino que tiene una forma indefinida a la que se la denominó Geoide. Dicha forma no se ha podido representar mediante ninguna fórmula matemática; es por esto que se ha seguido considerando a la tierra como un elipsoide. Estos elipsoides son definidos sobre una elipse que gira sobre el eje mayor. Se han estudiado muchas aproximaciones de la forma de la tierra mediante un elipsoide, cada una de ellas establece distintos valores para los parámetros de éste: excentricidad, semiejes, y focos. Algunos de las aproximaciones más conocidas son el WGS84, WGS79.

Sin embargo, debido a la falta de simetría mencionada anteriormente, existen grandes zonas del planeta que no responden a la curvatura de un elipsoide sino que son más bien planas o, incluso, presentan hundimientos. Alguno de los elipsoides

nombrados podrían aproximar muy bien una zona pero podrían no hacerlo con otras. Por lo tanto, distintas partes del planeta podrían aproximarse con distintas superficies.

Para localizar un elipsoide en un espacio se define un datum. Un datum es el elipsoide con su posición en el espacio. La posición esta definida por el origen y la orientación del elipsoide con respecto al geoide. Diferentes datums localizan latitud y longitud en diferentes posiciones en el espacio.

# 4 Sistemas móviles y sensibilidad al contexto

## *4.1 Introducción*

En el presente capítulo se describe el concepto de mobile computing y las tecnologías necesarias para su implantación, así como también, los sistemas basados en la ubicación, quienes agregan a los sistemas móviles la complejidad de brindar información o servicios de acuerdo a la ubicación del usuario que esta interactuando con el sistema.

Por último, se detallan las diferentes definiciones de contexto, especificando la aplicada en esta tesis, para luego describir los sistemas sensibles al contexto.

## *4.2 Mobile Computing*

En la presente sección se muestran las definiciones actuales sobre mobile computing, las implicaciones de trabajar con sistemas móviles y lo que para diferentes autores abarca la movilidad. Además se detallan problemas que deben enfrentarse al trabajar con sistemas móviles.

### 4.2.1 Definiciones

*A continuación se presentan algunas definiciones de "mobile computing"*

Según Zimmerman [Zimmerman, 99] el término "mobile computing" es usado para describir el uso de dispositivos de computación, los cuales generalmente interactúan de alguna manera con un sistema de información central, a pesar de que el lugar de trabajo no es fijo. La tecnología de mobile computing permite al trabajador móvil crear, acceder, procesar, almacenar y comunicar información sin restringirse a una única ubicación. Para extender el alcance de un sistema de información fijo de una organización, mobile computing permite la interacción con el personal de la organización que previamente estaba desconectado (cuando no se encontraba conectado al sistema en un lugar fijo de trabajo).

Según Forman et al. [Forman, 94] los avances en la tecnología inalámbrica han engendrado un nuevo paradigma de computación llamado mobile computing, en el cual los usuarios que portan dispositivos móviles tienen acceso a una infraestructura compartida independiente de su ubicación física. Esto proporciona comunicación flexible entre personas y acceso continuo a los servicios de la red.

Mobile Computing es una importante tecnología en evolución. Esta permite a los usuarios móviles comunicarse e interactuar con un sistema de información fijo, mientras cambia su ubicación física. Mobile Computing puede ser implementada usando muchas combinaciones de hardware, software y tecnologías de comunicación. Es una tecnología estratégica y versátil que provee calidad de información y accesibilidad, incrementa la eficiencia operacional, y realza la eficacia administrativa.

## *4.3 Desafíos en el trabajo con mobile computing*

Diseñar software para mobile computing trae aparejado una serie de desafíos extra que el diseño de software tradicional. Según Forman et al. [Forman, 94] muchos de los problemas extras que debe tratar mobile computing tienen que ver con tres de sus propiedades esenciales: comunicación, movilidad y portabilidad.

### 4.3.1 Comunicación inalámbrica

Mobile computing requiere comunicación de acceso inalámbrico. La comunicación inalámbrica presenta mayores obstáculos que la comunicación no inalámbrica, debido a que el ambiente interactúa con la señal, bloqueando su camino e introduciendo ruido y distorsión. Como resultado la comunicación inalámbrica es caracterizada por un menor ancho de banda, mayor cantidad de errores y desconexiones más frecuentes. Estos factores hacen que se incremente la latencia de transmisión, debido a que deben efectuarse retransmisiones.

La movilidad también aumenta las posibilidades de desconexiones dado que el usuario al moverse puede entrar en zonas donde no haya cobertura o haya interferencia. Otro problema típico con la movilidad y las redes inalámbricas es que el número de usuarios en una célula varia dinámicamente, si hay gran cantidad de usuarios móviles en una célula (puede darse el caso por ejemplo dentro de un acto o convención) esto podría provocar una sobrecarga.

Dadas las características antes mencionadas el diseñador debe tener en cuenta

- Desconexiones
- Bajo ancho de banda.
- Gran variabilidad en el ancho de banda.
- Redes heterogéneas
- Riesgos de seguridad

#### 4.3.1.1 Desconexiones

Los diseñadores deben tener en cuenta cuando se pueden pasar recursos por la red previendo que no se produzcan desconexiones. También es deseable hacer la aplicación que se ejecuta en el dispositivo lo más independiente posible de forma que pueda trabajar ante una desconexión. A su vez los diseñadores deben diseñar buenas interfaces de aplicaciones de modo de informar de forma correcta errores de desconexiones.

#### 4.3.1.2 Bajo ancho de banda

Debe observarse cuando el ancho de la red es apropiado para hacer envíos de información y debe también preverse que en conexiones inalámbricas no se tiene un buen ancho de banda, con lo cual la información a enviarse por la red debe ser siempre acotada en tamaño.

#### 4.3.1.3 Gran variabilidad en el ancho de banda

Los diseñadores de mobile computing tienen que tener en cuenta que el ancho de la red varía ampliamente. Esto es, en su mayoría, debido a que la cantidad de usuarios por célula varía constantemente. Una aplicación puede tomar tres políticas dada la variabilidad del ancho de banda: asumir que siempre hay alto ancho de banda (y enviar siempre la mayor cantidad de información), asumir que nunca hay buen ancho de banda (y enviar solo información reducida), o adaptarse al ancho de banda y enviar información dependiendo de esta.

Por ejemplo en una aplicación de monitoreo de vehículos, la alternativa de pensar que siempre hay un buen ancho de red enviara el mapa completo con todos los vehículos, la de pensar que siempre el ancho de red es malo solo una zona mínima del mapa que contiene el vehículo y la que se ajusta al ancho de banda, un mapa completo o uno de zona mínima ajustándose al ancho de banda disponible.

#### 4.3.1.4 Redes heterogéneas

Cuando un dispositivo cambia de una red a otra, debido a la movilidad, puede variar tanto la velocidad como los protocolos utilizados en la red anterior. Además, otra situación que puede darse es que un dispositivo tenga acceso a varias conexiones a la vez, dado que celdas lindantes pueden superponerse.

#### 4.3.1.5 Riesgos de seguridad

La seguridad de comunicación inalámbrica puede ser comprometida mucho más fácilmente, especialmente si la conexión se extiende sobre grandes áreas. Debido a esto deben incluirse en el diseño de mobile computing medidas de seguridad. Las medidas que deben tenerse en cuenta es encriptación de los datos, la cual puede ser realizada por software o por hardware.

### 4.3.2 Movilidad

La capacidad de cambiar de ubicación mientras se encuentra conectado a la red incrementa la volatilidad de la información. Ciertos datos que son considerados estáticos para las aplicaciones comunes son dinámicos para mobile computig. Un ejemplo de esto es una computadora en un lugar determinado puede referenciar siempre al mismo servidor más cercano, pero una aplicación corriendo en un dispositivo móvil necesita un mecanismo para determinar cual es su servidor. Como la dirección de la red cambia dinámicamente, esto afecta tanto a parámetros de configuración como a las respuestas de las consultas realizadas por el usuario.

#### 4.3.2.1 Cambios de direcciones

Como la gente se mueve, su computadora móvil usa diferentes direcciones de acceso a la red. Para soportar los cambios frecuentes de direcciones se han propuesto diferentes esquemas para "mobile IP". A continuación se describen algunos de los esquemas existentes:

- **Broadcast Selectivo**: Con el broadcast un mensaje es enviado a toda las células de la red, si el dispositivo móvil buscado se encuentra responde enviando su dirección IP. Esto es demasiado costoso si la red es grande. Por

lo cual se utiliza el broadcast selectivo el cual hace un broadcast solo a una parte de las células de la red, pero para esto debe saberse que el dispositivo solo podría encontrarse en ese conjunto de células.

- **Servicios centrales**: con servicios centrales las direcciones son mantenidas en una base de datos lógicamente centralizada. Cada vez que un dispositivo móvil cambia su dirección IP envía un mensaje de update a la base de datos.

- **Base de origen**: es esencialmente lo mismo que servicios centrales, solo que la base de datos se encuentra distribuida.

- **Punteros de redirección:** con los punteros de redirección, cada vez que un dispositivo móvil cambia su ubicación deja una copia de su dirección actual en donde estaba su vieja dirección IP. Cuando se desea comunicarse con el dispositivo se sigue este encadenamiento para llegar a su dirección actual. Para que no sea muy ineficiente por las re-direcciones, pueden hacerse actualizaciones para reflejar las direcciones más recientes.

#### 4.3.2.2 Información dependiente de la ubicación

Dado que las computadoras tradicionales no se mueven, gran cantidad de información, como nombre del servidor local, impresoras disponibles, y hora en la zona, son configuradas estáticamente. El desafió en este campo es obtener los datos apropiados de cada ubicación para su correcta configuración.

#### 4.3.2.3 Cambios de ubicaciones

Mobile computing engendra una nueva clase de movilidad que cambia mientras el usuario se mueve. Si un dispositivo móvil es equipado para encontrar el servidor más cercano para un servicio dado, en un corto plazo la ubicación puede cambiar y el servidor puede dejar de ser el más cercano.

### 4.3.3 Portabilidad

Cuando se diseña para mobile computing se debe prever que el diseño debe adaptarse a los distintos tipos de dispositivos existentes, los cuales varían en sus capacidades. Se deben tener en cuenta los siguientes puntos:

#### 4.3.3.1 Capacidades diferentes de procesamiento

Dadas las diferencias entre los dispositivos no puede suponerse que todos tienen suficiente capacidad de procesamiento para encriptación y desencriptación de datos, lo cual complica la aplicación de normas de seguridad.

#### 4.3.3.2 Energía Limitada

A diferencia de las computadoras de escritorio, los dispositivos móviles tienen energía limitada, por lo cual debe minimizarse el consumo de energía.

#### 4.3.3.3 Interfaz reducida

La interfaz en los dispositivos móviles es de tamaño reducido. Además no puede suponerse que todos los dispositivos brindan las mismas características de interfaces. Algunas de las consideraciones que deben evaluarse en el diseño de la interfaz son:

- La pequeña superficie que brinda un dispositivo móvil lleva la mayoría de las veces a sacrificar botones por campos de entrada. Estos campos de entrada pueden ser tomados por reconocimiento de escritura, de gestos, o de vos; pero esto hace que el sistema sea menos usable que con disponibilidad de botones, por lo tanto es muy importante evaluar cuales son las prioridades de la aplicación para el correcto desarrollo.
- El Mouse no se encuentra disponible en la mayoría de los dispositivos móviles, con lo cual debe cambiarse la forma estándar de diseñar interfaces a la que se esta acostumbrado en las computadores de escritorio.

#### 4.3.3.4 Capacidad de almacenamiento

La capacidad de almacenamiento en dispositivos portátiles es limitada. Tener discos de almacenamiento lleva a un mayor consumo de energía que dispositivos de memoria, con lo cual la mayoría de los dispositivos no poseen discos de almacenamiento. Debido a esto algunas de las soluciones planteadas son, compresión automática de archivos, acceso a almacenamiento remoto a través de la red, compartir librerías de código y compresión de páginas de memoria virtual.

## *4.4 Servicios basados en la ubicación*

A continuación se introducirán conceptos generales de lo que se conoce como servicios basados en la ubicación.

Un servicio basado en la ubicación o LBS es la capacidad de encontrar una ubicación geográfica de un dispositivo móvil y proporcionar servicios basados en la información de ubicación. Por ejemplo una persona en un shopping podría pedir los restaurantes cercanos económicos, y necesita solo saber los nombres y direcciones de aquellos que están dentro de sus límites, digamos 1 Km. a la redonda.

Según GSM [GSM, 03] los servicios basados en ubicación son servicios empresariales y de consumidores que da a los usuarios un conjunto de servicios comenzando desde la ubicación geográfica del cliente. Esos servicios ofrecen la posibilidad a los usuarios o maquinas de encontrar/ubicar otras personas, maquinas, vehículos, recursos y también servicios sensibles a la ubicación, así como también la posibilidad de rastrear su propia ubicación. El pedido de ubicación puede ser originado desde el cliente en si mismo o desde otra entidad como una aplicación proveedora o la red. De cualquier manera, cuando la ubicación es requerida, el cliente tiene que dar permiso al pedido de ubicación. Los servicios basados en ubicación cortan a través de varias clases de servicios móviles dado que agregan la característica de ubicación de éstos.
La mayoría de los servicios de ubicación móvil debe incluir dos importantes acciones:

1. Obtener la ubicación del usuario
2. Utilizar esta información para proporcionar un servicio

Los servicios basados en ubicación pueden ser automáticamente disparados cuando el cliente se encuentra en una ubicación específica, por ejemplo, áreas geográficas sujetas a diversos sistemas de publicidad. Alternativamente esta puede ser originada desde el cliente para satisfacer requerimientos basados en la ubicación tales como necesidades de información; encontrar puntos de interés, personas o chequear condiciones de tráfico, vehículos, recursos, servicios o requerimientos de máquinas y emergencias.

Para ValuedLBS [ValuedLBS, 04] LBS emplea el conocimiento de la ubicación del usuario para permitir la provisión de servicios nuevos o enriquecidos a un usuario vía dispositivos inalámbricos (wireless), por ejemplo, teléfonos móviles, PDAs, etc.

Las investigaciones de LBS se enfocan sobre que es lo que el usuario esta tratando de hacer, que información necesita y como agregar valor proveyéndoles información relevante a la ubicación y el tiempo (la información correcta en el momento correcto de la manera correcta).

LBS conoce donde esta el usuario, y así es capaz de proporcionar servicios que son relevantes a ese usuario y a esa ubicación. Cuando se habla de LBS, generalmente se refiere a usuarios móviles, es decir, servicios entregados al usuario vía un dispositivo inalámbrico tal como un teléfono móvil o PDA.

Los servicios móviles estándar (por ejemplo, noticias, clima, MMS – multimedia messaging service) no tienen en cuenta donde se encuentra el usuario. Los LBS son diferentes – porque ellos pueden determinar la ubicación del usuario (y podrían conocer la ubicación de otras personas que son importantes para él), ellos pueden ofrecer nuevos tipos de servicios o niveles enriquecidos de servicios para un cliente.

### 4.4.1 Ejemplos de LBS:

- **Directivas para encontrar un camino:** si solicita un servicio para encontrar un camino hasta un destino que esta tratando de alcanzar a pie, en auto o a través de otro medio, este servicio deberá ser capaz de ubicarlo, rastrear su progreso, y darle directivas hacia su destino. Esas directivas podrían ser una serie de instrucciones, mapas electrónicos o ambas.
- **Encontrar algo cercano a usted:** un LBS puede decirle donde se encuentra un restaurante italiano o cajero automático más cercano a su ubicación. Si la ubicación de esos lugares son conocidos por un servicio, dado que el conoce su ubicación, este puede resolver cuan distantes son dichos puntos de interés y darle directivas si es necesario.
- **Ubicando a su familia y amigos:** LBS no fue apuntado a un usuario individual - puede ser utilizado por una comunidad para permitir a otros usuarios saber donde usted está. ¡Por supuesto habrá veces en que usted no desee que otras personas sepan donde está!

Todo suena bueno en teoría, pero hay algunos desafíos al diseñar LBS exitosamente. Aquí están algunos ejemplos:

- **LBS depende de poder localizar al usuario con exactitud:** para algunos servicios (por ejemplo clima local), el posicionamiento celular es adecuado. Para servicios que ofrecen horarios de trenes a un conmutador, el posicionamiento celular puede probablemente calcular si usted esta en el trabajo o en su casa, y

dar los apropiados horarios de trenes. Para otros servicios tales como proporcionar instrucciones de navegación actualizadas dinámicamente a conductores, un posicionamiento más exacto tal como GPS (o mejoras a GPS) es necesario.

- **Se debe conocer donde las cosas están:** esto suena obvio, pero si un servicio le dice al cliente donde esta la estación de servicio más cercana, entonces el servicio debe conocer las ubicaciones de todas las estaciones de servicio.
- **Se necesita conocer exactamente que es lo que el cliente esta tratando de hacer:** por ejemplo, las personas a menudo tratan de hacer una combinación de cosas al mismo tiempo. Usted puede querer encontrar el restaurante más cercano, verificar el menú, reservar una mesa, pedir un taxi para tomarlo desde allí y conseguir algo de efectivo en el camino. Para mejorar la utilidad de los sistemas actuales, el siguiente paso es integrarlos de una manera significativa.

Para Whereonearth [Whereonearth, 04] los servicios basados en ubicación pueden ser descriptos como aplicaciones que reaccionan a cambios de ubicación geográfica. Un cambio de ubicación geográfico puede ser un evento como entrar a un edificio, la posición de un usuario móvil o la posición de un vehículo.

Según ellos usar el conocimiento de donde algo se encuentra o donde intenta ir es la esencia de los Lbs.

### 4.4.2 Componentes de LBS:

- **Geocoding:** es la tarea de procesar direcciones textuales para agregar una coordenada posicional (latitud-longitud) a cada dirección. Esas coordenadas son luego indexadas para permitir buscar las direcciones geográficamente, como por ejemplo "encontrar el punto más cercano" de manera más rapida.

- **Contenido de mapa:** puede estar en formato raster o vectorial. Las imágenes raster son imágenes pre-renderizadas (tal como fotos aéreas o el London A-Z) mientras que los datos vectoriales son una serie de capas coincidentes, cada capa contiene un tipo de información específica (parques, autopistas, calles, ríos, etc.). Ambos formatos pueden ser usados para mostrar mapas sobre una pantalla.

- **Búsqueda de proximidad:** es el método de detectar información "relevante" para encontrar los requerimientos específicos de usuarios. Ejemplos incluyen; "encontrar todo dentro de un radio de...", "seleccionar todo lo que conduciré en la próxima hora" o "mostrar donde estoy".

- **Ruteando y manejando direcciones:** es la interacción entre la ubicación de los usuarios (origen) y el destino planeado. Las rutas pueden ser calculadas y mostradas sobre un mapa y se pueden proporcionar directivas para la "distancia más corta" o la "ruta más rápida". En el caso de cambiar las situaciones del tráfico o camino, los datos del tráfico pueden ser mezclados con el contenido del mapa estático para proporcionar rutas alternativas en tiempo real y ajustar el tiempo de recorrido.

- **Rendering:** es la producción de mapas para mostrar sobre la pantalla de un dispositivo. Las imágenes son generalmente personalizadas de acuerdo al requerimiento LBS específico.

Según "Sun Developer Network Site" [SDNS, 04] los servicios basados en la ubicación responden 3 preguntas: donde estoy? Que esta alrededor mío? Como llego allí?. Estas determinan la ubicación del usuario mediante alguna tecnología para determinar la posición, luego usan la ubicación y otra información para proporcionar aplicaciones y servicios personalizados. Como ejemplo, consideremos un servicio de emergencia 911 inalámbrico que determina la ubicación del llamante automáticamente. Tal servicio podría ser extremadamente útil, especialmente para usuarios que están lejos de su hogar y no conocen su ubicación. Los asesores de tráfico, ayuda navegacional incluyendo direcciones y mapas, y ayudas en banquinas son servicios basados en la ubicación naturales. Otros servicios pueden combinar la ubicación actual con información sobre las preferencias personales para ayudar a los usuarios a encontrar comida, hospedaje, y entretenimiento acorde a sus gustos y su bolsillo.

Hay dos enfoques básicos para implementar servicios basados en la ubicación:

1. Procesar los datos de ubicación en un servidor y enviar los resultados al dispositivo.
2. Obtener los datos de ubicación de una aplicación basada en el dispositivo que la usa directamente.

Para descubrir la ubicación del dispositivo, LBS debe usar métodos de posicionamiento en tiempo real. La exactitud depende del método utilizado.

Las ubicaciones pueden ser expresadas en términos espaciales o como descripciones de texto. Una ubicación espacial puede ser expresada en el sistema de coordenadas latitud-longitud-altitud, donde la latitud es expresada de 0 a 90 grados norte o sur del ecuador, longitud de 0 a180 grados este u oeste del meridiano de Greenwich, y la altitud en metros sobre el nivel del mar. Una descripción de texto es generalmente expresada como una dirección de calle, incluyendo ciudad, código postal, y así sucesivamente.

Las aplicaciones pueden utilizar varios tipos de métodos de posicionamiento.

- **Usar la red de teléfonos móviles:** la "cell ID" actual puede ser usada para identificar la Estación Base Transmisora-Receptora (BTS, Base Transceiver Station) con la que el dispositivo se esta comunicando y la ubicación de tal BTS. Claramente, la exactitud de éste método depende del tamaño de la célula, y puede ser bastante inadecuado.
- **Usando satélites:** GPS utiliza 24 satélites que orbitan sobre la tierra. GPS determina la posición calculando las diferencias en los tiempos de las señales desde diferentes satélites al alcance del receptor. Las señales GPS son codificadas, así el dispositivo móvil debe estar equipado con un receptor GPS. GPS es el método más exacto (entre 4 y 40 metros de exactitud). Para más información sobre GPS ver sección 2.1.
- **Usando guías de posicionamiento de rango corto:** en áreas relativamente pequeñas, tal como una simple construcción, una red de área local puede proporcionar ubicaciones junto con otros servicios. Por ejemplo, dispositivos

equipados apropiadamente pueden usar Bluetooth para posicionamiento de rango corto.

Algunas aplicaciones no necesitan gran exactitud, pero otras serán inútiles si la ubicación no es bastante exacta.

### 4.4.3 Conclusión

Los sistemas de LBS son aplicaciones móviles que tienen en cuenta la ubicación para adaptar su comportamiento, en la sección 4.6 se verán los sistemas sensibles al contexto y podrá apreciarse que los LBS son sistemas que ofrecen servicios sensibles al contexto pero basado solo en la ubicación.

## *4.5 Tecnologías Móviles*

### 4.5.1 Introducción

El desarrollo de tecnologías de dispositivos móviles y comunicaciones inalámbricas ha aumentado considerablemente, y hoy en día son ampliamente difundidas en el mercado. Estos posibilitan acceso ubicuo[2] y sistemas de cálculo distribuido, así como también servicios y repositorios remotos. Esta sección describe en líneas generales las tecnologías existentes para los usuarios móviles actuales.

### 4.5.2 Dispositivos Personales

En el mercado de computadoras portátiles actualmente existe una amplia gama de dispositivos que van desde grandes notebooks (que ofrecen similares características que una computadora personal), a pequeños y discretos dispositivos como teléfonos celulares, que tiene capacidades de computo más limitado. El rango actualmente incluye:

- **Notebooks and Sub-notebooks:** las notebooks ofrecen funcionalidad similar a una PC. Las sub-notebooks ofrecen mayor portabilidad con insignificante reducción de la perfomance del sistema. Sin embargo, el resultado que esto acarrea es la reducción del tamaño de la pantalla [Sony].

- **Tabletas digitales** (Digital Tablets)**:** utilizan una pantalla touch-sensitive[3] y un lapiz digital para la entrada del usuario. Ofrecen la misma performance que las notebooks, con un incremento del tamaño de la pantalla, no poseen teclado y pueden tener los sistemas operativos y ambientes de desarrollo estándares.

- **Handheld y Palm PCs:** Handheld PCs (H/PCs) son pequeños y livianos dispositivos portátiles disponibles en una variedad de tamaños y formas, cada una con diferentes pantallas, periféricos y sistemas operativos.

- **Asistentes digitales personales** (PDAs)**:** fueron originalmente diseñadas para ser dispositivos muy pequeños, livianos y con limitadas capacidades de expansión, que solo eran capaces de ejecutar aplicaciones simples tales como

[2] **Ubicuo**: que esta presente en todo momento y en cualquier lugar.
[3] **Touch-sensitive**: dispositivo que permite al usuario interactuar con una computadora tocando un área sobre la superficie del dispositivo con un dedo, lápiz, u otro objeto.

calendarios y libretas de direcciones. No obstante la generación actual de PDAs, ofrecen una variedad de aplicaciones y soporte de conectividad. Ejemplos de aplicaciones incluyen versiones de web browser, clientes de email, multimedia players, etc.

- **Teléfonos celulares.** Hoy en día existen una amplia variedad de teléfonos celulares, muchos de ellos proveen servicios adicionales como email, accesos a internet, administradores de información personal, etc. [Nokia].

- **Wereable computing:** Exploran el potencial de los dispositivos que son inconcientemente portados y personales como la ropa o la bijoutery. Ofrecen formas más naturales de interfaces facilitando una amplia variedad de capacidades de comunicación entre humanos y computadoras. [Wereable Computing].

### 4.5.3 Comunicaciones

#### 4.5.3.1 Redes inalámbricas

Por red inalámbrica entendemos una red que utiliza ondas electromagnéticas como medio de transmisión de la información, que viaja a través del canal inalámbrico enlazando los diferentes equipos o terminales móviles asociados a la red. Hoy día una de las tecnologías más prometedoras y discutidas en esta década es la de poder comunicar computadoras mediante tecnología inalámbrica. La conexión de computadoras mediante "ondas de radio" o "luz infrarroja", actualmente está siendo ampliamente investigada. Las redes inalámbricas facilitan la operación en lugares donde la computadora no puede permanecer en un solo lugar. No se espera que las redes inalámbricas lleguen a remplazar a las redes cableadas, dado que estas ofrecen velocidades de transmisión mayores que las logradas con la tecnología inalámbrica.

Las tecnologías de comunicación inalámbrica pueden ser categorizadas de la siguiente manera.

- **Personal Area Network (PAN)**: una red de área personal, en relación a la red de computadoras, es el termino usado para representar la interconexión inalámbrica de dispositivos electrónicos personales tales como notebooks, teléfonos celulares, etc., usando tecnologías tales como Bluetooth [Bluetooth].

- **Local Area Network (LAN)**: es una red de computadoras que se encuentra en un área geográfica relativamente pequeña, tal como, una habitación, un edificio o grupo de edificios.

- **Metropolitan Area Network (MAN)**: son usualmente caracterizadas por muy alta velocidad de conexión usando cables de fibra óptica u otros medios digitales y son más grandes en espacio geográfico que local área networks. Por ejemplo, conectan toda una universidad o una ciudad.

- **Wide Area Network (WAN)** Una red de computadoras que se encuentran en un área geográfica relativamente grande. Las computadoras son conectadas a una amplia red a través de redes públicas, tales como sistemas telefónicos, aunque pueden ser conectadas a través de líneas rápidas o sistemas de satélites.

El rango de tecnologías disponibles puede ser agrupada en 2 clasificaciones distintas: infla-red (IR) y radio frecuencia (RF). La mayoría de las implementaciones IrDA ("Infrared Data Association") son punto a punto (Direct IR), permitiendo baja tasa de transferencia. Sin embargo, sistemas con transferencia mayor proveyendo transferencia bi-direccional también existen (Diffuse IR). Los sistemas IR son generalmente mejor aceptados en ambientes interiores ya que las tecnologías no pueden fácilmente penetrar objetos tales como paredes o vidrios y son altamente susceptibles a perdida de información. Además, la comunicación punto a punto hace difícil comunicarse con múltiples dispositivos simultáneamente y por consiguiente establecer redes ad hoc.

Para tratar el problema de usar IR en un ambiente externo varias tecnologías RF han sido desarrolladas, ya que estas tienen mayor capacidad de penetrar objetos como divisiones de oficinas y paredes. Hoy en día hay una gran variedad de tecnologías comerciales inalámbricas LAN disponibles. Ejemplos de ellas son Nokia Wireless Accelerator [Nokia, 1], Alvarion [Alvarion], Intel [Intel], etc.

Existen dos tipos de redes de larga distancia: Redes de Conmutación de Paquetes (públicas y privadas) y Redes Telefónicas Celulares. Estas últimas son un medio para transmitir información de alto precio. Debido a que los módems celulares actualmente son más caros y delicados que los convencionales, ya que requieren circuiteria especial, que permite mantener la pérdida de señal cuando el circuito se alterna entre una célula y otra. Otras desventajas de la transmisión celular son:

- La carga de los teléfonos se termina fácilmente.
- La transmisión celular se intercepta fácilmente (factor importante en lo relacionado con la seguridad).
- Las velocidades de transmisión son bajas.

Todas estas desventajas hacen que la comunicación celular se utilice poco, o únicamente para archivos muy pequeños como cartas, planos, etc.. Pero se espera que con los avances en la compresión de datos, seguridad y algoritmos de verificación de errores se permita que las redes celulares sean una opción redituable en algunas situaciones.

La otra opción que existe en redes de larga distancia son las denominadas: Red Pública De Conmutación De Paquetes Por Radio. Estas redes no tienen problemas de pérdida de señal debido a que su arquitectura está diseñada para soportar paquetes de datos en lugar de comunicaciones de voz. Las redes privadas de conmutación de paquetes utilizan la misma tecnología que las públicas, pero bajo bandas de radio-frecuencia restringidas por la propia organización de sus sistemas de cómputo.

### 4.5.3.2 Redes públicas de radio

Las ondas de radio pueden viajar a grandes distancias y penetrar los edificios sin problemas, razón por la cual se usan tanto en interiores como en exteriores. Las ondas de radio son omnidireccionales o sea viajan en todas las direcciones por lo que el transmisor y receptor no tienen que alinearse. Las propiedades de la onda dependen de la frecuencia. A bajas frecuencias las ondas de radio cruzan bien los obstáculos, pero la potencia disminuye drásticamente con la distancia de la fuente. A frecuencias altas, las ondas tienden a viajar en línea recta y a rebotar por los obstáculos, aunque también son absorbidas por la lluvia. En todas las frecuencias, las ondas de radio están sujetas a

interferencia por motores y otros equipos eléctricos. Esta es una de las razones por la cual, los gobiernos legislan el uso de los radiotransmisores. Las redes públicas tienen dos protagonistas principales: "ARDIS" (una asociación de Motorola e IBM) y "Ram Mobile Data" (desarrollado por Ericcson AB, denominado MOBITEX).

#### 4.5.3.3 Redes Infrarrojas

Las ondas infrarrojas se usan para comunicaciones de corto alcance y no atraviesan objetos sólidos, lo cual ofrece una ventaja de no interferencia. Además, la seguridad de los sistemas infrarrojos contra espionaje es mejor que la de los sistemas de radio, no es necesario obtener licencia del gobierno para operar un sistema infrarrojo. Las redes de luz infrarroja están limitadas por el espacio y casi generalmente la utilizan redes en las que las estaciones se encuentran en un solo cuarto o piso. Algunas compañías que tienen sus oficinas en varios edificios realizan la comunicación colocando los receptores/emisores en las ventanas de los edificios. Las transmisiones de radio frecuencia tienen una desventaja: que los países están tratando de ponerse de acuerdo en cuanto a las bandas que cada uno puede utilizar, al momento de realizar este trabajo ya se han reunido varios países para tratar de organizarse en cuanto a que frecuencias pueden utilizar cada uno. La transmisión infrarroja no tiene este inconveniente por lo tanto es actualmente una alternativa de red a tener en cuenta.

### 4.5.4 Tecnologías de censores

La existencia de tecnologías de censores de bajo costo permite a un sistema monitorear aspectos de su ambiente operativo o sus usuarios, esta situación provee una base para desarrollar aplicaciones móviles que pueden adaptarse a los cambios del contexto.

Las tecnologías de censores se pueden dividir en:

**Censores de ubicación**: Ejemplos son GPS y localización por las antenas de celulares.

**Censores de ambiente físico**: Ejemplos son dispositivos de censores de presión, temperatura, humedad, lluvia, etc. Algunos ejemplos son los productos de IButton [Ibutton], Pointsix [Pointsix].

**Censores de ambiente fisiológico**: monitorear el cuerpo humano puede proveer información relacionada al estado actual del usuario operante. Considerando ejemplos como equipos de vuelo de líneas aéreas, conductores de trenes o personal operante con maquinaria pesada, la operación segura de estos sistemas es dependiente de su personal y por lo tanto, puede beneficiar supervisar los factores fisiológicos que pueden ser perjudiciales a su operación segura.

## *4.6 Contexto y sensibilidad al contexto*

En esta sección se introducirá la idea de contexto, ya que para utilizarlo efectivamente, se debe tener un mejor conocimiento de lo que este es. Un mejor entendimiento permite a los diseñadores escoger que contexto usar en sus aplicaciones, proveer comprensión en los tipos de datos que necesitan ser soportados y las abstracciones y mecanismos requeridos para soportar computación sensible al contexto.

Luego se pasara a dar una definición de sistemas sensibles al contexto y las diferentes clases de adaptaciones posibles.

### 4.6.1 Introducción a Contexto

Ha habido numerosos intentos de definir contexto y sensibilidad al contexto, pero en general se han basado en ejemplos de contexto o vagas referencias a sinónimos de contexto relacionadas a la ubicación como se detallan posteriormente.

Uno de los primeros que introdujo la idea de context-aware, lo que en este trabajo se denomina "sensibilidad al contexto"4, fue el trabajo sobre Active Maps de Schilit y Theimer [SchilitTheimer, 94] refiriéndose al contexto como ubicación, identidad de la cercanía de personas y objetos, y los cambios que ocurren para esos objetos.

En una definición similar, Brown et al. [Brown, 96] en su trabajo sobre Stick-e Note Architecture define al contexto como ubicación, identidad de las personas alrededor del usuario, hora, estación, temperatura, etc.

Ryan et al. [Ryan, 98] define al contexto como la ubicación del usuario, el entorno o ambiente, la identidad y el tiempo.

Estas definiciones definen contexto con un ejemplo y son difíciles de aplicar en diferentes situaciones.

Otras definiciones simplemente han dado sinónimos de contexto, refiriéndose por ejemplo al contexto como entorno o situación. Algunas consideran el contexto como el entorno del usuario, mientras que otras lo consideran como el entorno de la aplicación. Brown [Brown, 96] definió al contexto como los elementos del entorno del usuario que la computadora del usuario conoce. Franklin y Flaschbart [FranklinFlaschbart, 98] ven esto como la situación del usuario.

Dey [Dey, 00] establece una definición abstracta en relación a su trabajo sobre herramientas de contexto:

*"Contexto es cualquier información que puede ser usada para caracterizar la situación de una entidad. Una entidad es una persona, lugar u objeto que es considerado relevante para la interacción entre un usuario y una aplicación, incluyendo al usuario y la aplicación."*

Schmidt et al. [Schmidt, 98] propone dos categorías para el contexto: los factores humanos y el entorno físico cada uno con tres subcategorías.

Los factores humanos son estructurados con las siguientes tres subcategorías:

- **Usuario:** Esta caracterizado por los hábitos del usuario, el estado mental o características fisiológicas.
- **Entorno social:** esta caracterizado por la proximidad de otros, por sus relaciones sociales y tareas colaborativas.

[4] Si bien difiere de la traducción literal que sería conocimiento del contexto, nos parece mas adecuada dado el uso que se le da en una aplicación al contexto que se conoce, es decir, esta varía dependiendo del contexto en el que no encontremos.

- **Tareas:** define las actividades dirigidas al objetivo o los objetivos generales del usuario.

El entorno físico esta estructurado en la siguientes tres subcategorías:

- **Ubicación:** esta podría ser absoluta (ej: una coordenada GPS), o relativa (ej: dentro de una habitación particular).
- **Infraestructura:** describe el ambiente informático y el entorno de interacción.
- **Condiciones:** Describe las condiciones físicas del entorno tales como ruido, claridad, presión, temperatura, etc.

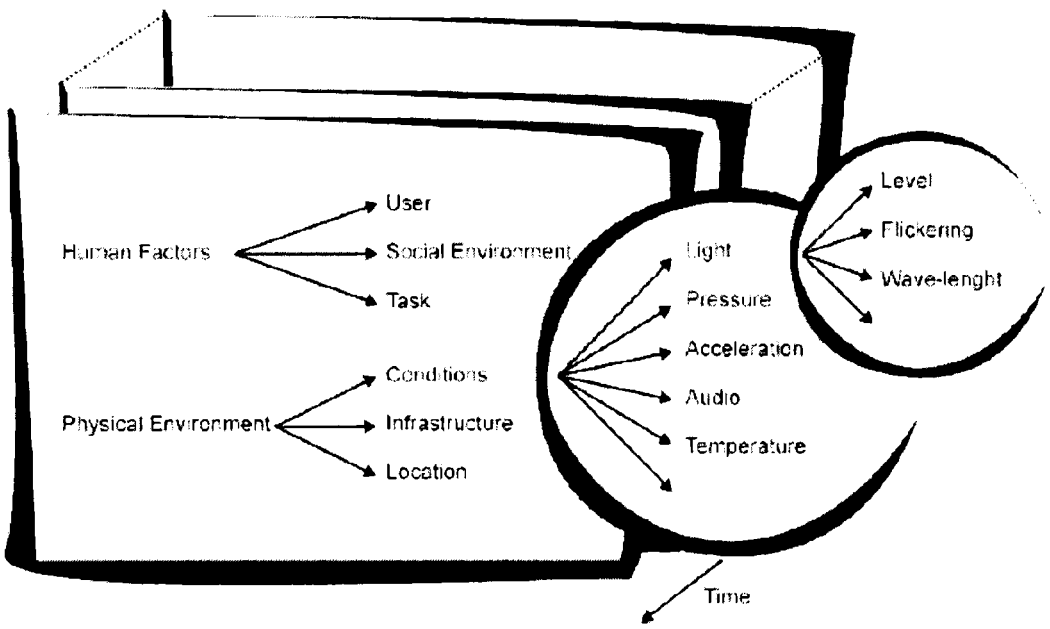


**Figura 4.1: Estructura del entorno físico**

Schmidt et al. [Schmidt, 98] también incluyen el tiempo como ortogonal a estas categorías, dado que el conocimiento de contextos anteriores puede proveer utilidad a una aplicación.

Dix et al. [Dix, 99] define una taxonomía para el contexto en términos de la interacción del humano con la computadora y sostiene que un dispositivo móvil opera en un amplio contexto que incluye la red y la infraestructura computacional, el amplio sistema computacional, el dominio de la aplicación y el entorno físico. La taxonomía puede ser resumida como se muestra en el siguiente cuadro:

| Contexto | Relaciones con | Problemas |
|---|---|---|
| Infraestructura | Ancho y disponibilidad de red, resolución de la pantalla | Variabilidad del servicio, conocimiento del usuario del servicio, tiempo de vida del dato |
| Sistema | Otros dispositivos, aplicaciones y usuarios | Aplicaciones distribuidas, tiempo de respuesta y retardo ocasionado por la infraestructura técnica, comportamiento emergente |
| Dominio | Dominio de aplicación, estilo de uso, identificación del usuario | Interacción con el dispositivo móvil, personalización, privacidad |
| Físico | Naturaleza física del dispositivo, entorno, ubicación | Naturaleza de la movilidad, información dependiente de la ubicación, uso de los sensores de entorno |

### 4.6.2 Sensibilidad al contexto (Context-Awareness)

Schilit y Theimer [SchilitTheimer, 94] fueron unos de los primeros en discutir sobre aplicaciones sensibles al contexto en 1994, es decir, software que se adapta de acuerdo a su ubicación de uso, la colección de objetos y personas cercanas, host y dispositivos accesibles, así como también los cambios de esas cosas en el tiempo. Para ellos hay tres aspectos importantes: donde esta, con quien esta, y que recursos están cercanos.

Mas tarde, Hull et al [Hull, 97] y Pascoe et al [Pascoe, 98] definieron a las aplicaciones sensibles al contexto más explícitamente como la capacidad de los dispositivos de computación de detectar y percibir, interpretar y responder a aspectos del entorno local del usuario y los dispositivos de computación en si mismos. Posteriormente, Dey et al [Dey, 00] en el año 2000 realiza su propia definición de context-awareness de la siguiente manera:

*"Un sistema es sensible al contexto si este usa el contexto para proporcionar información relevante y/o servicios al usuario, donde la relevancia depende de las tareas del usuario."*

Las definiciones en su categoría más específica de "adaptar el contexto" requieren que el comportamiento de una aplicación sea modificado para ser considerada context-aware. Cuando se intenta usar esas definiciones para establecer que aplicaciones son context-aware, estas no resultan adecuadas. Por ejemplo, una aplicación que simplemente muestra el contexto del entorno del usuario al usuario no esta modificando su comportamiento, pero es context-aware. Si usamos definiciones menos generales esas aplicaciones podrían no ser consideradas context-aware. Por este motivo, Dey escoge una definición más general e inclusiva que no excluye aplicaciones sensibles al contexto existentes y no esta limitada a las otras definiciones más generales.

Las características de las aplicaciones sensibles al contexto son generalmente categorizadas en alguna de las siguientes categorías:

1. presentación de información y servicios al usuario.
2. ejecución automática de un servicio.
3. etiquetar la información de contexto para luego recuperarla.

La presentación se refiere a aplicaciones que simplemente muestran información contextual o servicios al usuario, por ejemplo, una aplicación que muestra las coordenadas GPS por donde ella transita. Sin embargo, Dey et al [Dey, 00] afirma que muchas veces es difícil distinguir entre la presentación de información y la presentación de servicios. Por ejemplo, Schilit et al [SchilitTheimer, 94] demuestra que una lista de impresoras organizadas por proximidad al usuario es un ejemplo de proporcionar información al usuario, aunque esto puede depender de la intención de uso. Si el usuario solo mira la lista para familiarizarse con los nombres de las impresoras cercanas entonces se esta usando la lista como información. Pero, si un usuario elige imprimir con una de las impresoras de la lista entonces se está utilizando la lista como sistema de servicios próximos. Es por esto que información y servicios son tratados de manera similar para evitar intentar hacer suposiciones sobre el objetivo del usuario.

La ejecución automática (o auto ejecución) incluye acciones activadas por el contexto [SchilitTheimer, 94] y adaptación contextual [Pascoe, 98] y la capacidad de ejecutar o modificar un servicio automáticamente basado en el contexto actual. Ejemplos de aplicaciones sensibles al contexto que tienen este comportamiento puede ser encontrada en GUIDE [Davies, 98], una aplicación turística capaz de automáticamente mostrar páginas de información de turismo cuando los usuarios se acercan a atracciones particulares y agrega servicios sensibles al contexto al clipboard estándar de Windows basado en los datos copiados al clipboard.

Etiquetar la información de contexto para luego recuperarla o aumento contextual [Pascoe, 98a] es la capacidad de asociar datos digitales con el contexto del usuario. El proyecto forgetme- not de Xerox Research Centre Europe (XRCE) [Lamming, 94] (formalmente EuroPARC) es un buen ejemplo de una aplicación que toma ventaja de su tipo de comportamiento. En este proyecto Lamming et al. usan la idea de que el contexto físico puede ser un poderoso indicio para recordar eventos pasados, por ejemplo, cuando un usuario esta intentando recordar un documento particular que corregía tiempo atrás, puede no ser capaz de recordar el nombre del documento precisamente, pero será más probable recordar que el documento fue presentado en un encuentro particular, en un día particular, con ciertos colegas presentes.

### 4.6.3 Conclusión

Existen diversas definiciones de contexto y sensibilidad de contexto, pero en su gran mayoría estas se limitan a un ejemplo más que a una definición en si misma. Esto hace difícil determinar si cierta información es contexto y cuando una aplicación es sensible al mismo. Por este motivo en este trabajo se utilizarán las definiciones, tanto de contexto como de sensibilidad al contexto, formuladas por Dey, ya que son más inclusivas y permiten ser aplicadas en diferentes situaciones.

# 5 Descripción del sistema AVL y Usuarios móviles

## 5.1 Introducción

En este capítulo se detallan las características que posee el sistema AVL y los usuarios móviles. Este sistema esta construido para satisfacer los requerimientos de una aplicación típica de AVL y a su vez posee usuarios móviles que hacen requerimientos a ella. Aquí se detallan las funcionalidades de las aplicaciones de una central, que administra la información de todos los vehículos y brinda servicios, y de los usuarios móviles, quienes patrullan una ciudad y hacen requerimientos de información. Además se describen las adaptaciones que sufrirán las aplicaciones móviles, utilizadas por estos usuarios, en base al contexto.

## 5.2 Descripción global del sistema

El sistema consta de una central, la cual debe conocer información de cada vehículo e información cartográfica, y dos clases de vehículos: los monitoreados y los patrulleros.

La Figura 5.1 muestra las 3 componentes esenciales intervinientes en el sistema.

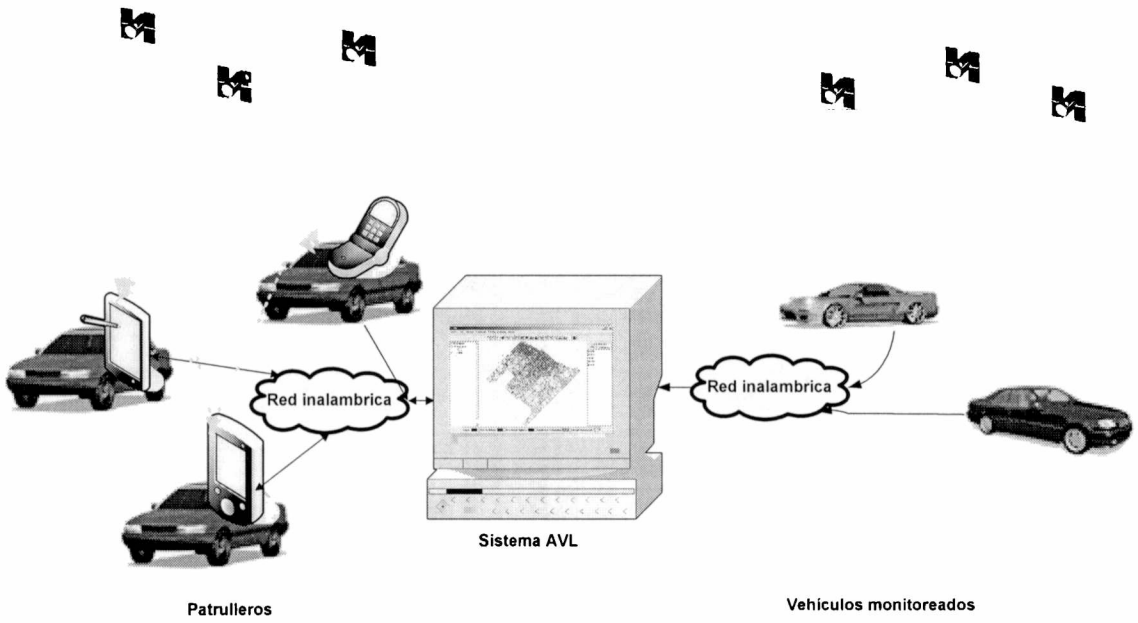


**Figura 5.1: Vista global del sistema**

Como puede observarse en la figura, los vehículos monitoreados toman su posición desde satélites GPS y mediante un dispositivo de comunicación la envían a la empresa de seguimiento que posee el sistema AVL. Esta información será utilizada para monitoreo y efectuar una recuperación en caso de robo, donde el robo puede ser declarado telefónicamente o activando una señal desde el dispositivo que se encuentra en el vehículo monitoreado.

La empresa de seguimiento podría brindar más de un tipo de servicio a quien la contrata (por ejemplo, no monitorear en ciertos horarios, en ciertas zonas monitorear

siempre, etc.), de esta manera, si se contrata más de un tipo de servicio puede optarse entre cual de ellos elegir en un momento determinado.

Los patrulleros son vehículos pertenecientes a la empresa que recorren la ciudad vigilando a los vehículos monitoreados, y en caso de que uno haya sido robado pueden asignarle la persecución del mismo. A su vez, al igual que los vehículos monitoreados, envían su posición al sistema AVL. Como puede verse en la Figura 5.1 contarán con algún tipo de dispositivo móvil que les permitirá manejar y visualizar cierta cantidad de información sobre el resto de los vehículos y datos del mapa en la que se encuentran.

## *5.3 La central*

### 5.3.1 Funcionalidades de la central

En esta sección se describen las funcionalidades necesarias para una central en las que desea hacerse el monitoreo de todos los vehículos existentes en el sistema AVL. Estas funcionalidades fueron obtenidas del relevamiento de sistemas AVL existentes y otras propuestas para esta tesis. A continuación se detallan las funcionalidades a tener en cuenta:

1) **Posicionar los vehículos en el mapa para visualizar la posición tanto de los vehículos monitoreados como de los patrulleros.**

   La central recibe información de donde se encuentran los vehículos monitoreados y los patrulleros, se encarga de almacenarla y mostrarla en una interfase para visualización de todos los vehículos (tanto monitoreados como patrulleros) del sistema.

2) **Asignar el robo de un vehículo a los usuarios móviles más cercanos al hecho.**

   Cuando un auto ha sido notificado como robado la central asigna un grupo de patrulleros para la recuperación del mismo dependiendo de cómo estén ubicados y el lugar en el que se encuentra el auto robado,. La forma de resolver como asignar patrulleros a un robo se hará en base a una estrategia de asignación, pudiendo eventualmente definir otras.

3) **Elegir que capas de información se desea visualizar y el orden de precedencia entre ellas.**

   La central maneja capas de información, por ejemplo, tiene información de calles, comisarías, hospitales, áreas de cobertura de comisarías, etc. Debe permitirse elegir la información y la prioridad de visualización de la misma.

4) **Determinar distancia entre puntos.**

   Dado dos puntos, se debe determinar la distancia existente entre ellos, pero esta distancia no será tomada linealmente sino que se tomara a través de un camino a seguir. Deberá elegirse entre diferentes caminos (más corto en distancia, más rápido, más barato, etc.).
5) **Información particular de vehículos y sus posibles conductores.**

Una vez que un vehículo particular ha sido seleccionado, se suministrará la información existente de este y sus posibles conductores con su respectiva información.

6) **Búsqueda de vehículos por diferentes criterios.**

El sistema proveerá funcionalidad de búsqueda de vehículos por diferentes criterios. **Algunos criterios de** búsqueda de vehículos son: dueño del vehículo, posibles conductores., patente, etc..

7) **Ingresar, eliminar y modificar información de vehículos y datos relacionados a ellos en el sistema.**

El sistema permitirá que se ingresen nuevos autos para su monitoreo. Estos serán dados de alta con los datos del vehículo, del dueño, del cliente que paga el servicio, y de sus posibles conductores. Esta información a su vez podrá ser modificada o eliminada del sistema.

8) **Seguimiento detallado de vehículos.**

Además de la visualización de todos los vehículos monitoreados, el sistema permitirá seleccionar un conjunto de vehículos y obtener una vista en el mapa de solo estos. En el caso de que solo un vehículo sea seleccionado se incluye una vista de toda su información.

9) **Suministrar la información de un punto (Ej. comisarías, hospitales, etc.).**

Dado un punto del mapa se proveerán todos los datos de interés que se encuentren en éste. Por ejemplo, si en el punto se encuentra un hospital se brindará la información conocida del mismo.

10) **Suministrar información de camino entre un punto y otro, teniendo en cuenta camino más corto, camino más rápido, etc., incluyendo información de distancia, y tiempo del camino.**

Cuando se solicita la búsqueda de un camino entre dos puntos se deberá elegir el criterio por el que desea hacer. Los criterios serán: camino más corto, camino más rápido (el cual debería tener en cuenta los datos de demora en las calles y velocidad máxima permitida), camino más económico (el cual debería considerar datos tales como cobro de algún peaje y cantidad de combustible necesario para llegar).

11) **Información de tráfico y retardos en un camino.**

Dado un camino, se puede pedir información adicional actualizada de este. Esta puede ser de tráfico, la cual indica la cantidad aproximada de vehículos en cada tramo del camino (un tramo puede ser una cuadra, un puente dentro de una ciudad, o una delimitación de kilómetros en un ruta), o de retardos, la cual establece las demoras aproximadas en dichos tramos.

12) **Informar la recuperación de un vehículo a los usuarios móviles involucrados en la persecución.**

Cuando un auto ha sido recuperado de un robo, la central avisa a los patrulleros avocados a la recuperación del vehículo que este ya ha sido recuperado, para que pase ahora a la tarea de vigilancia o eventualmente a dedicarse a la persecución de otro vehículo robado.

13) **Asignar o desasignar un patrullero a la persecución de un robo**

El sistema debe permitir seleccionar patrulleros y asignarlos o desasignarlos a un robo, de manera de proporcionar intervención humana.

14) **Funcionalidades de manejo de visualización de mapas como zoom, pan etc.**

Se deben brindar un conjunto de funcionalidades de manejo de mapa, entre las que se encuentran

- Mayor y menor nivel de zoom.
- Movimiento a través del mapa (pan).
- Vista inicial.
- Vista completa.
- Vista anterior y posterior
- Movimientos al norte, sur, este u oeste.

### 5.3.2 Servicios de la central

Además de las funcionalidades existe un conjunto de servicios que debe brindar la central a los usuarios móviles y a los vehículos monitoreados. Estos serán provistos por diferentes motivos, ya sea porque no tienen esa capacidad de cómputo, porque no ha recibido la información necesaria para la acción, o porque debe enviarle la información de su contexto a la central. De las funcionalidades anteriormente mencionadas existe un grupo que también serán suministradas como servicios.

A continuación se detallan los servicios propuestos:

1) **Recepción de posicionamiento de vehículos monitoreados.**

Cada cierto intervalo de tiempo definido, los vehículos monitoreados enviarán información de su ubicación. Dicha información será una o varias coordenadas por las que el vehículo se desplazo y su identificador.
El intervalo de tiempo dependerá del tipo de servicio que el vehículo haya contratado y del estado en el que se encuentra (robado o no).

2) **Información de posicionamiento de un vehículo particular.**

Ídem a la funcionalidad (1) descripta anteriormente.

3) **Información de una zona a patrullar (este servicio es suministrado a usuarios móviles que no tienen suficiente capacidad de procesamiento, Un**

**ejemplo sería solicitar el nombre de una calle dado que en la escala del mapa no se puede apreciar).**

Los servicios de información detallada de una zona son:

- Información de nombre, altura, dirección y demás información de nuestra base de datos correspondiente a las calles.
- Información de lugares particulares como hospitales, comisarías, etc.

4) **Información particular de vehículos y sus posibles conductores.**

Ídem a la funcionalidad (5) descripta anteriormente.

5) **Camino y distancia a un punto, teniendo en cuenta camino más corto, camino más rápido, etc., incluyendo información de distancia, y tiempo del camino.**

Ídem a la funcionalidad (10) descripta anteriormente.

6) **Información de tráfico y retardos en un camino.**

Ídem a la funcionalidad (11) descripta anteriormente.

7) **Elegir que capas de información se desea visualizar**

Ídem a la funcionalidad (3) descripta anteriormente.

8) **Informar la recuperación de un vehículo a los usuarios móviles involucrados en la persecución.**

Ídem a la funcionalidad (12) descripta anteriormente.

9) **Asignar o desasignar un patrullero a la persecución de un robo**

Ídem a la funcionalidad (13) descripta anteriormente.

### 5.3.3 Configuraciones de la central

Existe además un conjunto de parámetros a configurar en la central, donde algunos de ellos serán tomados por quienes se conecten a la misma.

1) **Tiempo de actualización de los datos en situación vigilancia.**

Dependiendo del servicio contratado se debe poder configurar con que frecuencia se actualizan los datos en situación de vigilancia.

2) **Tiempo de actualización de los datos en situación persecución (debería tener mayor frecuencia que la anterior).**

Se debe poder configurar la frecuencia en que se actualizan los datos en una persecución.

3) **Tiempo de almacenamiento del recorrido de los vehículos.**

Dependiendo del servicio contratado será el tiempo que se guardarán los datos de los caminos seguidos por los vehículos monitoreados en el sistema. Estos tiempos pueden ser configurados para cada servicio en particular.

4) **Selección de estrategia para asignación de vehículos a zonas.**

Como se verá en la sección 6.2 "Un modelo para AVL" no existe una sola manera de asignar la zona a un vehículo. Desde la central podría seleccionarse que estrategia va a utilizarse.

5) **Selección de estrategia para asignación de patrulleros a recuperación de vehículos robados.**

Nuevamente como se verá en la sección 6.2 "Un modelo para AVL" no existe una sola manera de asignar vehículos a la persecución de un vehículo robado. Desde la central podría seleccionarse que estrategia va a utilizarse en todos los casos.

## *5.4 Vehículos monitoreados*

Los vehículos monitoreados no son usuarios móviles ya que no tienen ningún tipo de funcionalidad, solo deben tener la lógica necesaria para cambiar el tiempo de envió de su información de ubicación de acuerdo a su estado. Desde este punto de vista la información a tener en cuenta es tan solo su ubicación y estado (Robado o seguro, si no se encuentra robado).

Por lo tanto las únicas acciones que tiene el vehículo monitoreado son las siguientes:

1) **Enviar información de posición a la central.**

El vehículo monitoreado enviará la información de su posición cada un intervalo de tiempo definido dependiendo de su estado (robado o no) y tipo de servicio (se refiere a la calidad del servicio, preferencia del usuario, etc.).

2) **Enviar información de que ha sido robado.**

Cuando se produzca un robo el usuario del vehículo podrá presionar una alarma que indicará el intento de robo del vehículo y a partir de ese momento el automóvil pasará al estado de robado.

## *5.5 Usuarios móviles (patrulleros)*

Un requerimiento clave es tener en cuenta la movilidad de los patrulleros, más específicamente ver su comportamiento y necesidades. Los patrulleros se encuentran monitoreando zonas de una ciudad verificando que no ocurran robos en ella, en un instante de tiempo pueden recibir la información de que ha ocurrido un robo y deben perseguir al auto robado para recuperarlo. Esto significa que la aplicación móvil debe estar preparada para cambiar su comportamiento cuando un vehículo es asignado a una persecución.

Para poder proveerle facilidad en la persecución de vehículos, el dispositivo debe poseer una interfaz gráfica capaz de mostrar mapas digitales.

La aplicación debe proporcionar la flexibilidad suficiente para permitirle al usuario móvil utilizar la aplicación mientras conduce su vehículo, debido a esto el sistema debe ser "inteligente" como para facilitarle algunas tareas teniendo en cuenta su entorno.

Es por esto que es sumamente útil que la aplicación sea sensible al contexto del usuario de forma que facilitará la interacción con el sistema. Por ejemplo, cuando un vehículo se encuentra persiguiendo un auto que ha sido robado, realizar la actualización de la posición de este en forma casi instantánea para efectuar la persecución de manera adecuada, además que muestre los restantes patrulleros que se encuentran avocados a la persecución para que puedan encontrar un modo de rodear el vehículo robado.

En esta sección se detallan las funcionalidades que el usuario móvil tendrá disponibles y posteriormente se desarrollará como se adaptarán en base a la información contextual.

1) **Enviar su información de contexto para que la central determine que datos enviarle.**

   Cada cierto intervalo de tiempo establecido, el usuario móvil enviará su información de contexto actual a la central, esta información incluirá la posición, datos del dispositivo utilizado, ancho de banda de la red, entre otras; y la central determinará en ese instante cuales son los datos que deben enviarse.

2) **Solicitar información de un vehículo particular.**

   Además de la visualización de todos los vehículos monitoreados en la zona correspondiente al usuario móvil, se permitirá seleccionar un vehículo y obtener una vista en el mapa de este vehículo con su información particular.

3) **Solicitar el camino a un punto, incluyendo en el pedido de información que tipo camino se quiere, más corto, más rápido, etc.**

   Ídem a la funcionalidad (10) de la central descripta anteriormente.

4) **Solicitar información de retardos, y tráfico en un camino**

   Ídem a la funcionalidad (11) de la central descripta anteriormente.

5) **Solicitar la desasignación de persecución de un robo**

   Ídem a la funcionalidad (13) de la central descripta anteriormente.

6) **Solicitar capas de información a visualizar**

   Ídem a la funcionalidad (3) de la central descripta anteriormente.

7) **Funcionalidades de manejo de visualización de mapas como zoom, pan etc.**

Ídem a la funcionalidad (14) de la central descripta anteriormente.

## *5.6 Decisiones de diseño*

El diseño de la aplicación móvil trae aparejado una serie de desafíos extras debido a la movilidad. En esta sección se detallan una serie de decisiones que debieron tomarse para poder diagramar luego una arquitectura para el sistema.

### 5.6.1 Alternativas de almacenamiento de información en el usuario móvil

La información que debe conocer un patrullero no es solo la ubicación de los demás vehículos, sino también información relacionada a la cartografía para poder ubicar los vehículos en un mapa y visualizarlos.

Existen dos alternativas para almacenar esta información:

1) Que el usuario móvil posea toda la información cartográfica.
2) Que la información cartográfica sea enviada con la restante información de los vehículos.

La alternativa 1 tiene como desventaja que el dispositivo que posee el patrullero debe tener gran capacidad de almacenamiento, ya que debe poder contener la cartografía de toda la zona a monitorear.

La alternativa 2 tiene como desventaja que hay que transmitir grandes cantidades de información por la red.

Dadas las características de los dispositivos móviles actuales, quienes poseen capacidades de almacenamiento y procesamiento limitadas, y que solo es necesario el envío de la zona pertinente a monitorear por el patrullero, se elige la alternativa 2 como la adecuada a la situación actual.

### 5.6.2 Alternativas para la comunicación del usuario móvil

Como se mencionó anteriormente debe existir una comunicación entre el usuario móvil y la central. Es claro que el usuario móvil realizará una comunicación con la central cuando tenga requerimientos para efectuar, pero existen alternativas a tener en cuenta en relación a las actualizaciones de la información contemplada por los usuarios móviles. Para realizar esta comunicación que involucra actualizaciones pueden darse las siguientes alternativas:

1) La central informa a los usuarios móviles cada cierto tiempo los cambios ocurridos.
2) El usuario solicita a la central los cambios ocurridos cada un intervalo de tiempo.

Para este problema en particular es conveniente la utilización de ambas alternativas, esto se debe a que los usuarios móviles deben ser notificados inmediatamente de eventos como la asignación o desasignación a la persecución de un vehículo robado, por lo que la central debe poder informarle a los usuarios móviles

dichos cambios. Los restantes pedidos de actualizaciones serán solicitados por los usuarios móviles cada un intervalo de tiempo estipulado.

## *5.7 Contextos*

Como ya fue planteado con anterioridad la aplicación móvil que posee el patrullero es sensible al contexto, por este motivo se debe manejar información sobre el contexto del usuario. En esta sección se detalla la información contextual que fue considerada apropiada manejar para los usuarios móviles. Las propiedades de contexto a tener en cuenta son:

### 5.7.1 Estado de patrulleros:

Un patrullero puede encontrarse en dos situaciones diferentes:

- Persecución.
- Vigilancia.

El estado de vigilancia es cuando el patrullero se encuentra recorriendo la ciudad monitoreando los vehículos, cubriendo la zona que le fue asignada.

El estado de persecución se da cuando el patrullero se encuentra persiguiendo un vehículo que ha sido reportado como robado, intentando, posiblemente junto a otros patrulleros, la recuperación del mismo.

En el momento que se informa el robo de un vehículo, se le asigna al/los patrullero/s más cercano/s al robado la persecución del mismo, pasando a estos al estado de persecución. En el momento que el auto es recuperado o se descarta la posibilidad de recuperarlo un patrullero vuelve al estado de vigilancia.

Un criterio adoptado es que solo es posible asignar la persecución de un vehículo a un patrullero cuando se encuentra en un estado de vigilancia.

### 5.7.2 Localización:

La localización tendrá en cuenta la ubicación del usuario móvil dentro del mapa que representa la ciudad o ruta en la que se encuentra.

### 5.7.3 Red

Esta propiedad básicamente determina el ancho de banda de la red en un momento determinado.

### 5.7.4 Dispositivo

Esta propiedad contextual considera la siguiente información:

1) Resolución de la interfaz gráfica.
2) Cantidad de colores de pantalla.
3) Capacidad de procesamiento.
4) Capacidad de almacenamiento.

Dado las características del sistema se supone que el dispositivo usado por el usuario móvil tiene características gráficas y, procesamiento y almacenamiento

aceptables para el manejo de gráficos. Además se supone que posee la suficiente capacidad de almacenamiento para guardar al menos el mapa actual a visualizarse.

## *5.8 Reglas de adaptación en base al Contexto*

Dado que se trabaja con una aplicación sensible al contexto, la información enviada a los usuarios móviles debe ser adaptada de acuerdo al contexto que la influye. En esta sección se describen las adaptaciones propuestas para la aplicación móvil de acuerdo a las diferentes propiedades del contexto.

### 5.8.1 Reglas de adaptación en base a la localización y estado del patrullero:

#### 5.8.1.1 Tamaño de la zona de vigilancia

Cuando un patrullero envía su posición a la central ésta debe determinar la zona que debe monitorear dicho patrullero, en el caso de que el mismo se encuentre patrullando.

El tamaño de la zona informada puede variar por diferentes criterios y debe permitirse elegir con que criterio trabajar en cada momento. Los diferentes criterios por los cuales puede hacerse la asignación de zonas serán explicados en detalle en el capítulo 6 sección 2. Una estrategia posible podría tener en cuenta la cantidad de vehículos que se encuentran patrullando y la cercanía entre los mismos, lo cual permite que cada patrullero se encargue de una zona diferente.

Cuando se envía información de la zona la cantidad de datos suministrados puede variar dependiendo del tamaño de la zona. A su vez, el tamaño del área de cobertura de la zona modifica el nivel de detalle suministrado. Por ejemplo, para un tamaño de zona chico podría informarse las comisarías, hospitales, obstáculos, etc. de la zona, en cambio, si el tamaño fuese grande mostrarlas complicaría la visualización debido a la escala del mapa.

#### 5.8.1.2 Tamaño de la zona de persecución

Si un usuario móvil (patrullero) se encuentra persiguiendo a un vehículo robado solo se debe brindar la zona que involucra dicho patrullero, el vehículo robado y los otros móviles involucrados en la recuperación. De esta manera se reduce la cantidad de información transmitida y el usuario tiene presente solo la información necesaria para la persecución, permitiéndole ver los demás patrulleros y realizar un cercado del vehículo sustraído.

La figura 5.2 muestra la visualización adecuada de un mapa en una persecución

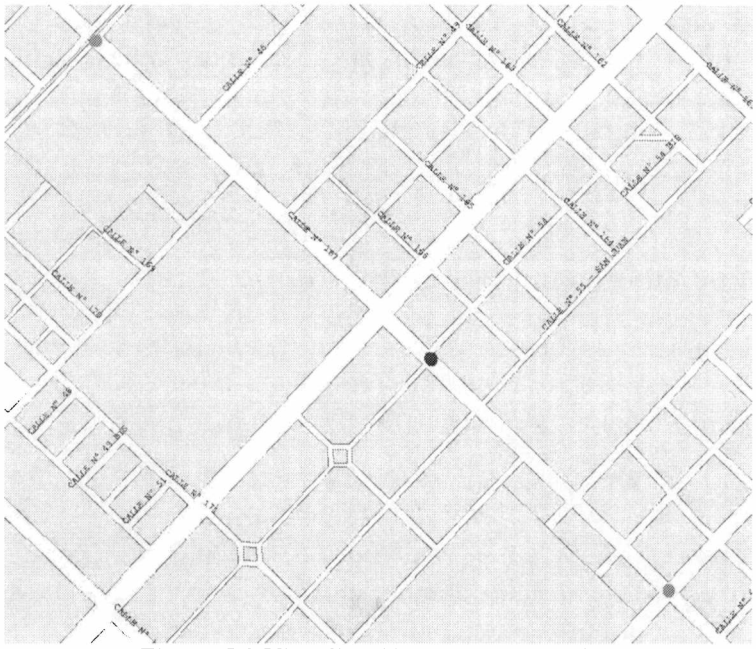

**Figura 5.2 Visualización para persecución**

En la figura pueden observarse dos patrulleros (en verde) dedicados a la persecución de un vehículo robado (en rojo).

### 5.8.1.3 Tipos de referencia de ubicación

Dependiendo del lugar donde se encuentre el patrullero se pueden tener diferentes formas de representar la ubicación de los vehículos que están siendo controlados.

Por ejemplo, si se encuentra en una ciudad la forma más común de determinar la ubicación es a través de la calle y altura de la misma, o indicar la calle en la que se encuentra acotándola entre las dos calles perpendiculares. La Figura 5.3 ejemplifica las visualizaciones apropiadas en una ciudad.

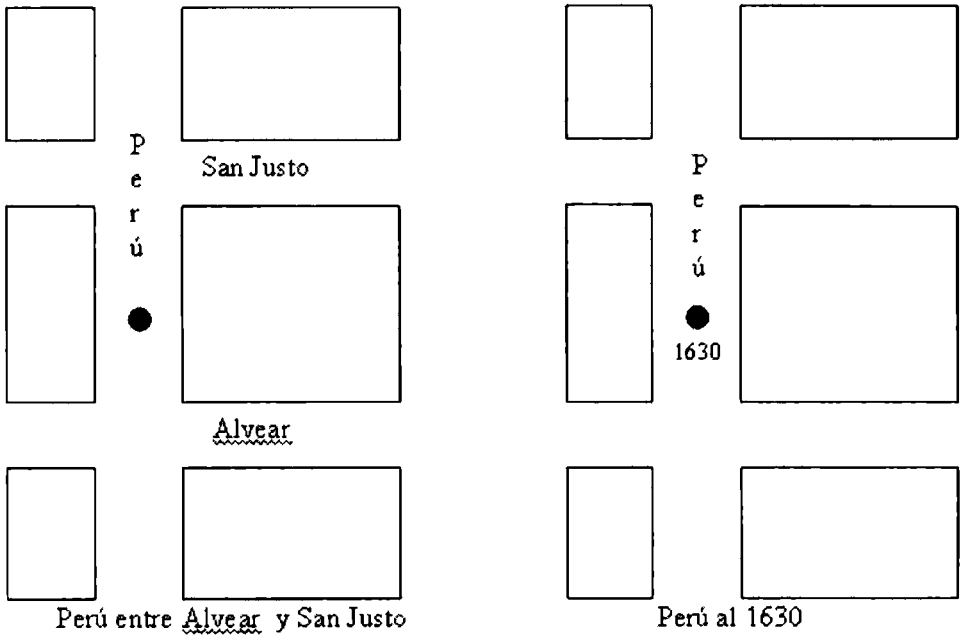


**Figura 5.3: Ubicación en una ciudad.**

Cuando el patrullero se encuentra en un puente, indicar la ubicación de ésta manera no sería correcto, por esto, solo se debería indicar el puente y el sentido de desplazamiento del vehículo. Por último, en el caso de encontrarse en una ruta se debería indicar la ruta, kilómetro dentro de la misma y sentido de desplazamiento del vehículo.

### 5.8.2 Reglas de adaptación dependiendo de la red

Antes de indicar las adaptaciones realizadas teniendo en cuenta la propiedad de contexto de red, se establecen asunciones en cuanto al servicio de red necesario para que dicha aplicación funcione correctamente. Estas son:

- El sistema debe ser capaz de soportar la caída de la red, pudiendo volver a solicitarse o enviarse la información requerida.
- Dada la poca información enviada por un vehículo monitoreado no necesita un ancho de banda demasiado grande, en cambio, para un patrullero en donde la cantidad de información transmitida en ambas direcciones es mayor se asumirá contratado un servicio de red con una tasa de transferencia mínima aceptable.
- Dada la naturaleza del servicio no deberían ocurrir caídas de la red por períodos tiempo de larga duración (mayores a 10'), si bien debería ser soportado por el sistema.

Asumido que estas restricciones son contempladas se discutirá a continuación las adaptaciones consideradas.

Dado que solo es considerado el ancho de banda de la red, dependiendo de este deberá variar la cantidad de información suministrada. Es decir, con un ancho de banda bajo, se enviarán menos capas (layers) de información de la zona. La central debería permitir configurar que capas de información enviar a los usuarios móviles para cada rango de ancho de banda en particular.

### 5.8.3 Reglas de adaptación en base al dispositivo

Dado que la propiedad de contexto de dispositivo tiene en cuenta varias características del mismo, a continuación se describe que adaptaciones son consideradas para cada una de ellas.

- Resolución de pantalla: dependiendo de la resolución de la interfaz gráfica (en píxeles y colores soportados) variará la calidad del mapa enviado al usuario móvil.
- Capacidad de procesamiento: si el dispositivo posee una alta capacidad de procesamiento este puede realizar ciertas operaciones sobre el mapa sin necesidad de requerírselas a la central, por este motivo se le debe proporcionar la mayor cantidad de información posible al mismo. Por ejemplo, podría realizar operaciones de zoom, pan, búsquedas de caminos entre dos puntos (no puede asegurarse que sea el mejor), etc., siempre y cuando estos datos estén comprendidos en la porción del mapa contenida en el dispositivo. Caso contrario, dichas operaciones deberán realizarse en la central.
- Capacidad de almacenamiento: dependiendo de la capacidad de almacenamiento podrán guardarse vistas anteriores de un mapa, de forma que la operación de ver

algunas de las vistas anteriores o posteriores se haga directamente en el usuario móvil y evitar requerirlas a la central.

En el caso de las dos últimas adaptaciones el usuario que se encuentra utilizando la aplicación no notará un cambio en el comportamiento de la misma, aunque es posible que observe un incremento de velocidad si ciertas operaciones se realizan en el dispositivo, dado que no es necesaria la demora ocurrida por la transmisión de datos a través de la red.

# 6 Descripción de diseño

## *6.1 Introducción*

En el presente capítulo se describe el diseño para el sistema y los usuarios móviles. Este comienza con la descripción de un modelo de base para una aplicación AVL. Luego, pasan a detallarse soluciones de diseño conocidas para aplicaciones sensibles al contexto y la solución propuesta en esta tesis. Para finalizar el capítulo se detalla la arquitectura utilizada para la central y los usuarios móviles, así como también el mecanismo de comunicación que existirá entre ellos.

## *6.2 Un modelo para AVL*

Como se mencionó en el capítulo 2 el modelo AVL debe proveer cierta funcionalidad básica. Para proveer dicha funcionalidad se debe administrar información sobre los vehículos involucrados en el sistema. Para administrar esta información, necesaria para un sistema AVL, necesitamos contar con al menos una central que provea servicios a los usuarios móviles, así como también de vehículos que suministren la posición en la que se encuentran, ya sean tanto vehículos monitoreados como patrulleros (usuarios móviles).

Por este motivo la funcionalidad de un sistema AVL debe ser distribuida entre los diferentes módulos que la componen. En este caso las partes que componen el sistema son: la central, los usuarios móviles (quienes solicitan servicios a la central), los vehículos monitoreados, y una capa de comunicación entre usuarios y la central.

Además de contener información y ubicación de los vehículos involucrados, el sistema debe proveer cierta funcionalidad, la cual será detallada a lo largo de este capítulo.

### 6.2.1 Introducción para un modelo básico

En un sistema de Localización Automática de Vehículos (AVL) es necesario manejar la información de los vehículos participantes junto con su posición.

Un vehículo contiene información que lo identifica unívocamente del resto de los vehículos existentes, junto con un conjunto de operaciones que permiten manipular sus datos e interactuar con el mismo, por ejemplo registrar sus recorridos, estimar su velocidad, etc.

Los vehículos pueden ser categorizados de dos formas:

- Vehículos monitoreados
  Un vehículo monitoreado es aquel que contrato un servicio a la aseguradora, registra periódicamente (cada intervalos de tiempo definido) la posición en la que se encuentra, conoce quienes son sus posibles conductores y puede estar en uno de dos estados posibles: robado o seguro (no robado). En el caso de que su estado sea robado conocerá información del robo pertinente.

- Patrullero (usuario móvil)
  Un patrullero modela un vehículo de la aseguradora encargado de vigilar y perseguir, en el caso de robo, los vehículos monitoreados que contrataron dicho servicio. Un patrullero es conducido por una persona y tiene dos posibles estados: vigilancia, cuando se encuentra recorriendo su zona asignada, y persecución cuando se encuentra asignado a la recuperación de un vehículo robado.

A su vez, el sistema de seguimiento debe ser notificado cuando un vehículo ha sido robado y, a partir de esto, disparar una persecución sobre el vehículo sustraído. Para esto se debe modelar la información del robo realizado a un vehículo monitoreado, junto con la fecha y hora del hecho, los patrulleros encargados de su recuperación y el recorrido realizado por el vehículo desde el momento de su robo hasta su recuperación, en el caso de que se logre, o se descarte la posibilidad de recuperarlo.

La recuperación refleja el hecho de que un vehículo robado ha sido recuperado por la empresa, conteniendo la fecha y hora de la recuperación, el robo y los patrulleros que lo lograron.

Mantener esta información permite realizar estadísticas tales como el porcentaje de recuperaciones realizadas por la empresa sobre el total de robos declarados, cantidad de recuperaciones realizadas en un período, patrullero que realizó más recuperaciones en un período, etc.

En la Figura 6.1 se muestra un diseño OO que modela lo descripto.

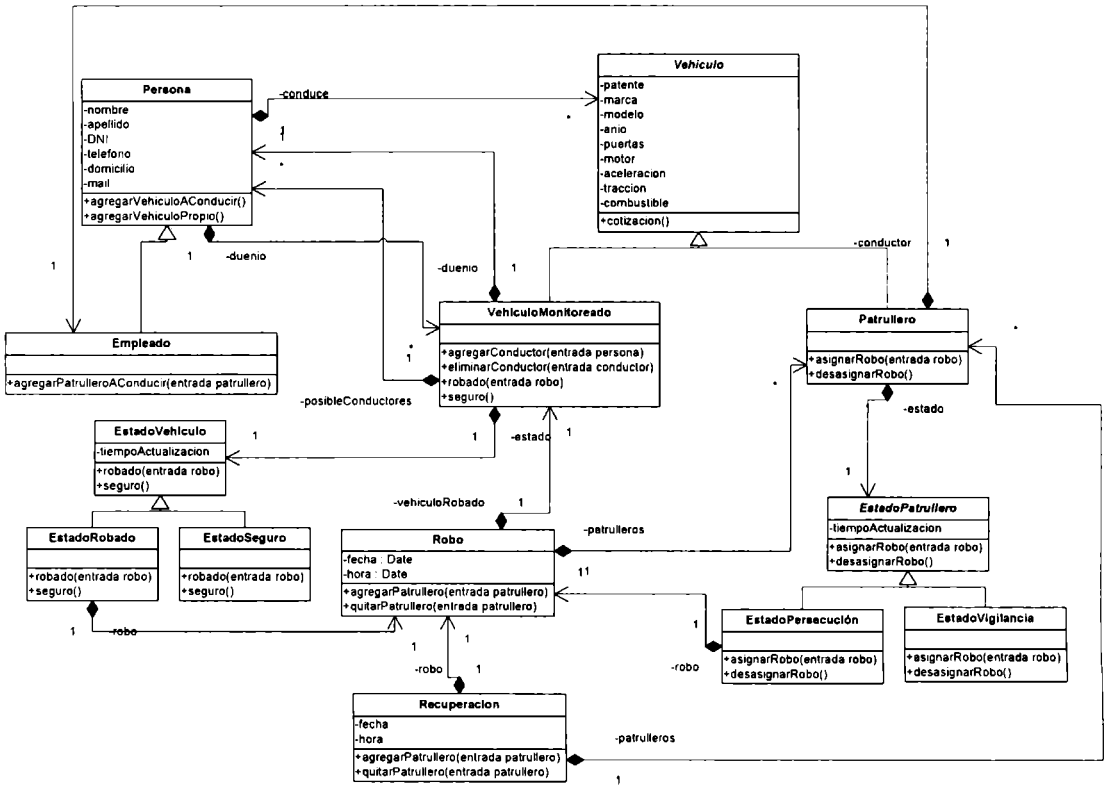


**Figura 6.1: Modelo básico de vehículos.**

El modelo presentado muestra la clase vehículo la cual contiene toda la información relevante a un vehículo. Como fue mencionado anteriormente existen dos clases de vehículos, los VehiculosMonitoreados y los Patrulleros, ambas subclases de Vehículo, quienes heredan su estructura y comportamiento.

Los VehiculosMonitoreados conocen su dueño y una lista de posibles conductores del mismo, además de su estado actual que puede ser uno de dos posibles, robado o seguro. Para modelar su estado se hizo uso del patrón de diseño State [Gamma, 95], donde dependiendo del estado se actuará de manera diferente ante la asignación o recuperación de un robo. Cuando un vehículo monitoreado sea robado se invocará al método robado(robo) y éste le delegará a su estado actual la responsabilidad de tomar la acción apropiada esperando recibir el estado apropiado si es que este cambia. En el caso de que el vehículo se encuentre en EstadoSeguro el método robado(robo) retornara el estado EstadoRobado como nuevo estado del vehículo conteniendo el robo asignado. Por el contrario, si el vehículo se encuentra en el estado "EstadoRobado" la asignación del robo no tendrá efecto, puesto que no tiene sentido declarar a un vehículo robado cuando éste ya lo esta. De la misma manera, cuando se invoque al método seguro() de un vehículo monitoreado este delegará dicha responsabilidad a su estado esperando por su nuevo estado apropiado. Al enviárselo a un EstadoRobado retornará un nuevo EstadoSeguro y no tendrá efecto en el caso del EstadoSeguro quien se retornará a si mismo.

Los Patrulleros conocen a su conductor actual (el cual es un empleado de la empresa aseguradora representada) además del estado en el que se encuentra, el cual puede ser uno de dos posibles: EstadoPersecucion y EstadoVigilancia. Cuando a un patrullero se lo asigna a un robo este se lo delega a su estado quien es el responsable de tomar la acción adecuada. Estos estados también son una aplicación del patrón de diseño State [Gamma, 95], donde ambos responden al método asignarRobo(robo) y desasignarRobo(). En el caso de EstadoDePersecucion se asigna el nuevo robo para que se comience la persecución sobre el vehículo sustraído en dicho robo, retornándose a si mismo, y en el de EstadoVigilancia retorna un EstadoDePersecución con el robo asignado.

De la misma manera cuando a un patrullero se le envía el método desasignarRobo(), este lo delega en su estado invocando el método desasignarRobo, el cual sobre un estado de persecución retorna un estado de vigilancia y en el caso de EstadoVigilancia no tiene efecto, pues el patrullero ya se encuentra en dicho estado, por lo cual el estado se retorna a si mismo.

El modelo también contiene las clases Robo y Recuperación que representan la información requerida con anterioridad. En el caso de un robo: el vehículo robado, la fecha y hora en que sucedió y los patrulleros encargados de su recuperación; y en el de una Recuperación el robo asociado, la fecha y hora en que se logró y los patrulleros que efectivamente lograron recuperarlo. Dada esta diferencia entre los patrulleros asignados al robo y los que realmente lograron recuperar el vehículo usurpado es que la Recuperación no los conoce a través del robo y posee su propia colección de patrulleros.

Por otro parte, el modelo contiene una clase Persona con los datos esenciales para representarla (como son su DNI, nombre, apellido, teléfono, domicilio y mail), junto con los vehículos que este conduce o posee. Dado que solo algunas personas pertenecen a la empresa, es decir son empleados y pueden conducir algún patrullero de la misma, es que se realiza una especialización de Persona con la clase Empleado,

heredando su estructura y comportamiento, y agregando conocimiento y comportamiento relacionado a los patrulleros que este puede conducir.

### 6.2.2 Modelando un mapa para AVL

Otra parte importante de un modelo AVL es la representación del mapa de la ciudad o ruta en la que se encuentran los vehículos participantes.

Un mapa de estas características esta formado por un conjunto de tramos, donde cada tramo puede continuar por uno o más de estos. Por ejemplo, una ciudad esta compuesta por calles, cada calle no necesariamente es un tramo sino que puede estar compuesta de varios tramos. Cada tramo representa una cuadra de la calle, y en sus extremos generalmente tiene más de una opción por la cual seguir. De la misma manera una ruta esta compuesta por diferentes tramos y puede continuar por una o más rutas a través de un empalme.

Un tramo debería poseer un nombre que lo represente (nombre de calle o ruta), una altura, una longitud, una unidad de medida (metros o kilómetros) y una posición (Location). A su vez, un tramo puede contener una serie de obstáculos que demoren el paso por el mismo o no permitan el paso a través de él. Ejemplos de estos son lomas de burro, semáforos, etc., los cuales demoran el paso por el tramo; y obras en construcción, accidentes sobre el tramo, etc., que impiden el paso momentáneamente.

Por otro lado, un mapa puede contener puntos de interés, como hospitales, comisarías, estaciones de bomberos, etc., en diferentes lugares del mismo.

La Figura 6.2 muestra un mapa de ciudad con sus calles, puntos de interés y obstáculos como fueron mencionados.

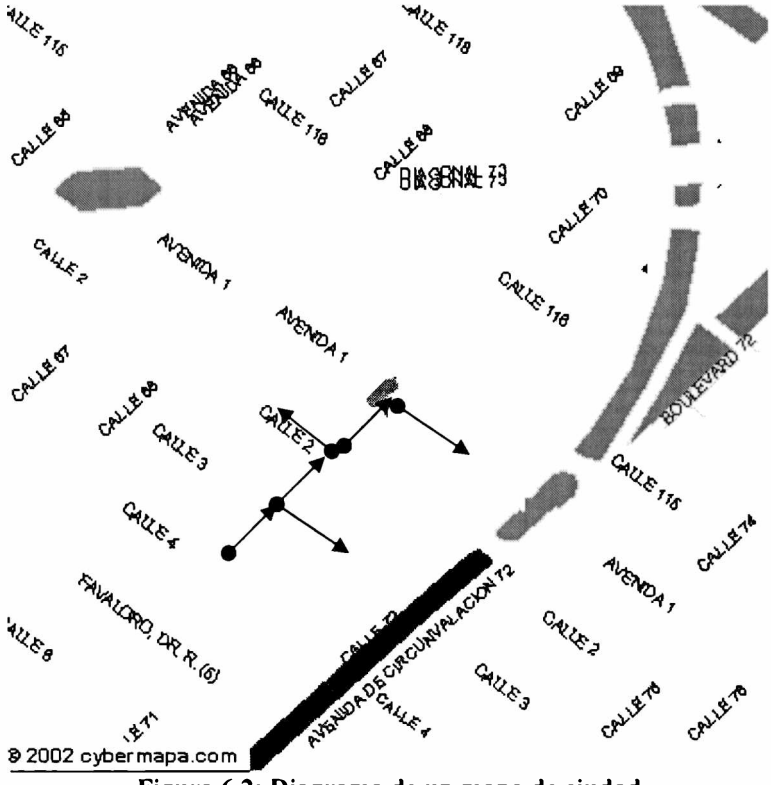


**Figura 6.2: Diagrama de un mapa de ciudad**

En la figura pueden verse las calles, donde las flechas ejemplifican como se va armando el grafo dirigido que las representa. También puede verse un punto de color naranja que simboliza que en este lugar se encuentra un hospital, y sobre la avenida 1 puede apreciarse que existe una loma de burro marcada con color marrón.

La Figura 6.3 muestra un modelo para representar el mapa junto con la información relacionada antes mencionada.

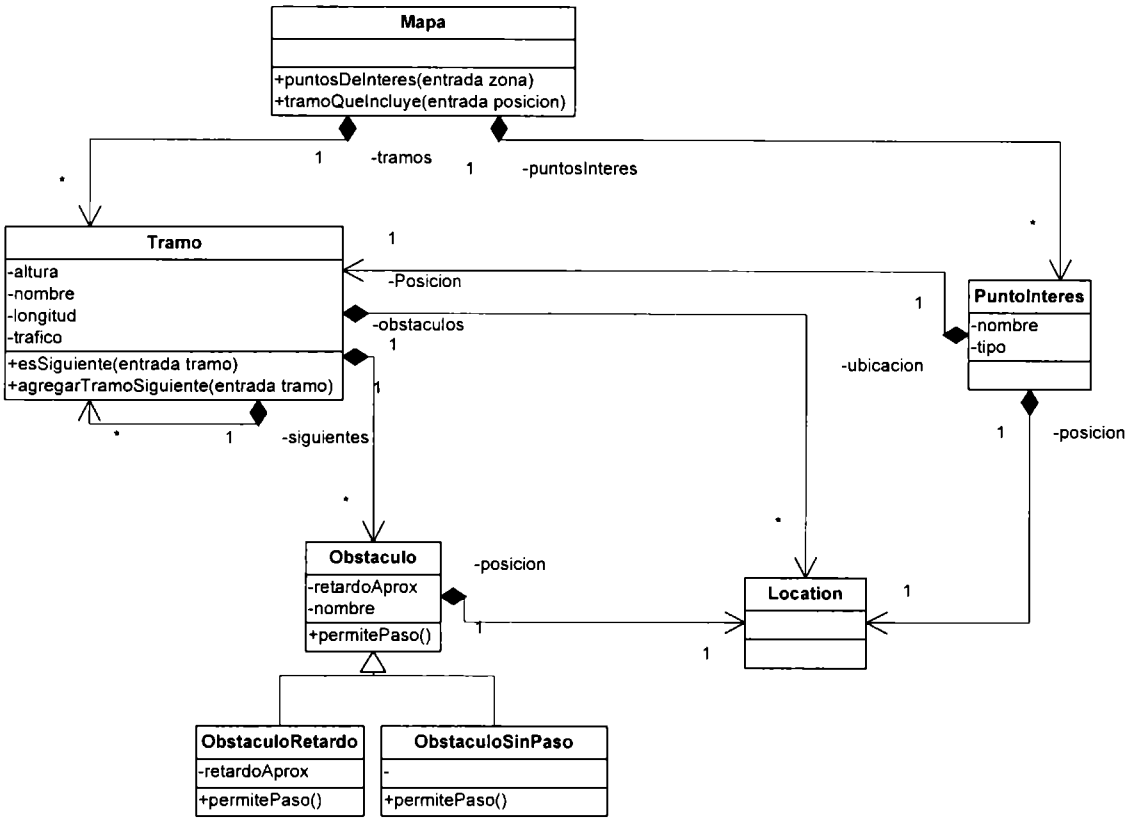


**Figura 6.3: Un mapa para AVL**

En el modelo la clase Mapa esta compuesta de una serie de tramos (instancias de la clase Tramo), donde cada uno conoce cuales son los tramos que pueden continuar a partir de él. De esta manera, el mapa forma un grafo dirigido donde cada tramo representa un vértice del mismo y el atributo siguientes la colección de arcos a partir de dicho vértice. Así, se logra un modelado de un mapa que permite representar tanto ciudades como rutas.

Además de la información propia que identifica un tramo este puede contener una serie de obstáculos que interrumpen o retardan el transito por el mismo. Dado que básicamente existen estos dos tipos de obstáculos y ambos deberían poder determinar si es posible cruzarlo, los obstáculos conforman una jerarquía de clases donde todos poseen un nombre (por ej., loma de burro, semáforo, obra en construcción, etc.) y saben responder al mensaje permitePaso(), que será primordialmente usado en la búsqueda de caminos para determinar si el tramo puede ser utilizado o no, detallado posteriormente. Los obstáculos que permiten el paso son modelados con la clase ObstaculoConRetardo y traen aparejado un retardo aproximado en segundos, y aquellos que no lo permiten se representan con la clase ObstaculoSinPaso.

Con la clase PuntoDeInteres se modela información de lugares útiles para el usuario AVL que se esta desplazando por la ciudad o ruta representada. Esta clase contiene un nombre, para determinar el nombre del punto de interés; un tipo, para establecer una distinción de la clase del punto de interés como puede ser comisarías, estaciones de bomberos, hospitales, estaciones de servicios, etc.; y el tramo en el cual se encuentra ubicado.

Tanto los tramos, los obstáculos, como los puntos de interés, necesitan una ubicación precisa dentro del mapa, es por esto que todas conocen su posición.

Para la representación de la posición se considera el trabajo de Gordillo et al. [Gordillo, 99] en el cual se describen aplicaciones GIS que constan de elementos georeferenciados quienes poseen una posición, interpretada en un marco geográfico o sistema de referencia, y una forma o topología.

### 6.2.3 Determinando recorridos para los vehículos

Como ya fue mencionado, un vehículo debe poder registrar los recorridos efectuados por el mismo. Un recorrido debe modelar el trayecto realizado por el vehículo. Los eventos que determinan el inicio o fin de un recorrido son el arranque y detenimiento completo del vehículo respectivamente, o el cambio de su estado.

El cambio de estado de vehículo genera un nuevo recorrido dado los siguientes casos:

- En un vehículo monitoreado el cambio de seguro a robado hace que se inicie un nuevo recorrido, puesto que evidentemente el recorrido que se encontraba haciendo el dueño del vehículo será interrumpido por el robo y llevado a uno nuevo por los delincuentes que lo sustrajeron. Un caso similar ocurre cuando el estado cambia de robado a seguro.
- En un Patrullero el cambio de estado de vigilancia a persecución interrumpirá un trayecto usual de un vigilante a otro en el que se realiza la persecución del vehículo robado. Un caso similar ocurre cuando el estado cambia de persecución a vigilancia.

En el caso de que un vehículo monitoreado sea robado, el robo deberá reflejar el recorrido realizado por el vehículo desde el momento en que se detecta el robo hasta que es recuperado o se descarta la posibilidad de lograr dicha recuperación.

Para lograr esto, un recorrido debe constar de las ubicaciones por las que se desplazo el vehículo, junto con la fecha y hora correspondiente. De esta manera, el recorrido podrá determinar la velocidad aproximada a la que se desplazo el vehículo, el tiempo consumido para llegar de un punto a otro, sentido del recorrido (en el caso de calles o rutas doble mano), etc.

Dado que sería poco práctico mantener un registro de todas las posiciones recibidas del vehículo, debido a que algunas de ellas no serían relevantes para el sistema o el cliente que contrato un servicio de cobertura no desea que se haga un seguimiento intensivo de todos sus recorridos, sería óptimo poder poseer diferentes estrategias de almacenado de dichos recorridos y poder seleccionar el que se creyera conveniente para cada caso particular.

A su vez, pueden existir diferentes criterios para almacenar los recorridos, como por ejemplo guardar solo los 5 últimos recorridos, los del último mes, todos, etc.

En la siguiente figura se presenta un modelo que refleja lo descripto.

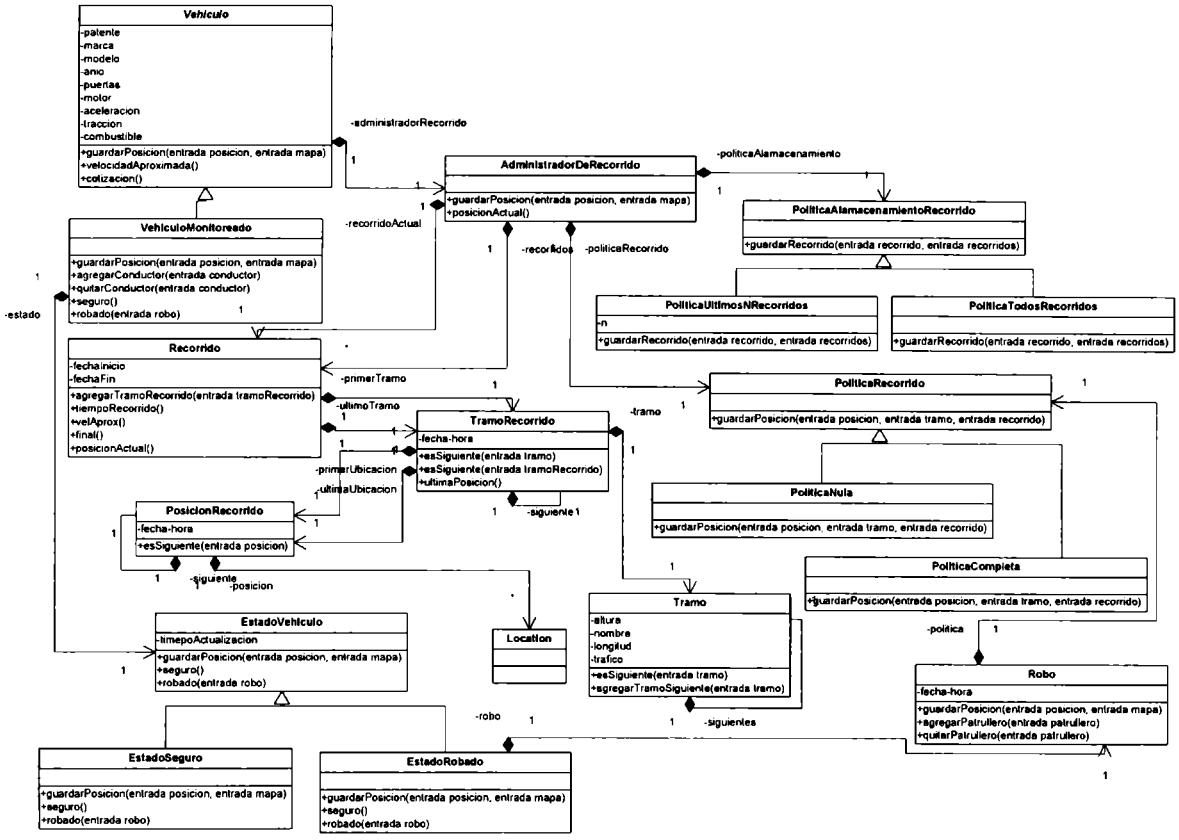


**Figura 6.4: Manejo de recorridos en AVL**

El vehículo conoce un AdministradorDeRecorrido, quien es el encargado de manejar toda la lógica asociada a las posiciones por las cuales se desplazo el vehículo. Modelado de esta forma se logra desacoplar la responsabilidad del manejo de recorridos del vehículo. La clase Vehículo tiene el mensaje guardarPosicion(posición, mapa), el cual delega la responsabilidad de guardar la posición en su manejador. El método guardarPosicion implementa la funcionalidad de guardar la posición recibida como parámetro en el recorrido actual, recibe además el mapa para obtener el tramo en el cual esta contenida dicha posición. Cuando el vehículoMonitoreado se encuentra robado, además de guardar la nueva posición en su recorrido actual, debe enviarle al robo asociado tal posición para que este la almacene en el recorrido realizado por el vehículo desde el momento que ha sido notificado su robo, es por esto que reimplementa el método guardarPosicion(posición, recorrido) y lo delega a su estado enviándole el mensaje guardarPosicion(posición, mapa) quien realizará las tareas relevantes para dicho estado.

Antes de mencionar como trabaja el AdministradorDeRecorrido detallamos como se modela un recorrido.

A través de la clase Recorrido se modela un recorrido realizado por un vehículo contenido en el sistema. Cada recorrido consta de una secuencia de tramos por la que se desplazo el vehículo en cuestión, pudiendo contener más de una posición dentro de dicho tramo, por lo cual, para estimar velocidad y saber en donde esta o estuvo el vehículo de manera precisa en diferentes momentos, se necesita asociar una fecha y hora a cada posición recorrida. Cada tramo recorrido es modelado con la clase

TramoRecorrido, el cual tiene asociado el tramo del mapa correspondiente por el que se desplazo el vehículo, y posee una secuencia de ubicaciones recorridas con su fecha y hora correspondiente. Para modelar estas posiciones recorridas aparece la clase PosicionRecorrido que registra la ubicación (Location) y la fecha y hora del paso por la misma. Tener varias ubicaciones dentro de un tramo recorrido es necesario puesto que por ejemplo, en tramos muy extensos o tramos de rutas que pueden constar de varios kilómetros, permiten tener una ubicación más precisa de donde se encuentra el vehículo en cada instante y determinar de forma más exacta consultas tales como velocidad promedio del vehículo, etc.

Dado que tanto los tramos recorridos como las ubicaciones son una secuencia es práctico conocer cual es el primer y último elemento dentro de la secuencia. El primero, básicamente, para saber como inicia dicha secuencia y a partir de él seguir por cada elemento correspondiente; y el último para poder agregar un nuevo elemento a la secuencia (que será el último de la misma) fácilmente.

Como fue mencionado el AdministradorDeRecorrido es el encargado de manejar todos los recorridos realizados por un vehículo, es por eso que conoce tanto una colección de recorridos realizados como el recorrido actual, y es este último quien puede responder la posición actual del vehículo. El administrador posee dos colaboradores donde cada uno tiene responsabilidades diferentes:

- Política de recorrido: quien determina que posiciones son relevantes para almacenar en el recorrido actual.
- Política de almacenamiento de recorrido: encargada de determinar cuales recorridos serán almacenados por el manejador en el momento de que el recorrido actual finaliza.

Ambas políticas fueron diseñadas como una aplicación del patrón de diseño Strategy [Gamma, 95], lo cual permite implementar diferentes variantes de cada política respectivamente, redefiniendo el o los métodos necesarios.

La política de recorrido es modelada a través de la clase PoliticaDeRecorrido quien declara la interfase común para los algoritmos soportados. En este caso, el método que debe ser reimplementado por las subclases es guardarPosicion(posición, tramoMapa, recorridoActual). El diseño muestra dos políticas por defecto: la política "PoliticaNula", la cual solo mantiene la posición actual del vehículo, es decir la última recibida; y una política "PoliticaCompleta" la cual registra cada una de las posiciones recibidas. Pueden incorporarse nuevas estrategias subclasificando la clase "PoliticaDeRecorrido" e implementando el método guardarPosicion(posicion, tramoMapa, recorridoActual).

La política de almacenamiento de recorridos es representada a través de la clase PoliticaAlmacenamientoRecorrido, quien define el método guardarRecorrido(recorrido, recorridos) en su interfase. Este método debe ser reimplementado por las subclases para definir nuevas estrategias de almacenamiento. En el modelo se proponen dos estrategias, la política PoliticaUltimosNRecorridos, a la cual se le debe configurar la cantidad N y solo almacenará los últimos N recorridos realizados por el vehículo con el que se esta trabajando; la política PoliticaTodosRecorridos almacena todos los recorridos. Es posible agregar nuevas políticas de almacenaje subclasificando la clase PoliticaDeAlmacenamientoDeRecorridos e implementando el método guardarRecorrido(recorrido, recorridos).

### 6.2.4 Incorporando Manejo de Zonas

Teniendo en cuenta que una compañía aseguradora posee varios patrulleros, es adecuado pensar en dividir el mapa en diferentes zonas, donde cada uno de los patrulleros será responsable de alguna zona individualmente o en conjunto con otros patrulleros. Esta división permite que los patrulleros estén esparcidos en el mapa y no concentrados en algún lugar dejando desprotegidos ciertos lugares del mapa (ciudad o ruta).

Cuando un patrullero se encuentra en estado de vigilancia posee una zona asignada, la cual es una subdivisión del mapa.

Realizar la división del mapa en zonas no es algo trivial, y no existe solo una forma hacerlo. Por ejemplo, la Figura 6.5 muestra en forma gráfica cómo sería una posible subdivisión del mapa en zonas fijas.

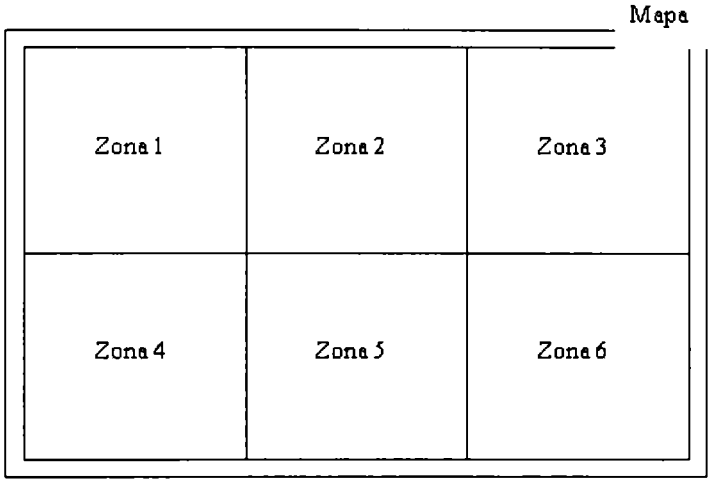


**Figura 6.5: División del mapa en zonas fijas**

Podría ocurrir que alguna zona, por ejemplo la 2, tuviera una mayor densidad de tránsito y por lo tanto hubiera más cantidad de vehículos monitoreados que transiten por allí. De esta manera, la división simétrica que muestra la Figura 6.5 no sería adecuada para un mapa con estas características si se quisiera tener una política de división que tenga en cuenta la densidad de transito para la asignación de patrulleros.

La Figura 6.6 muestra una división más adecuada para el mapa descripto en el párrafo anterior

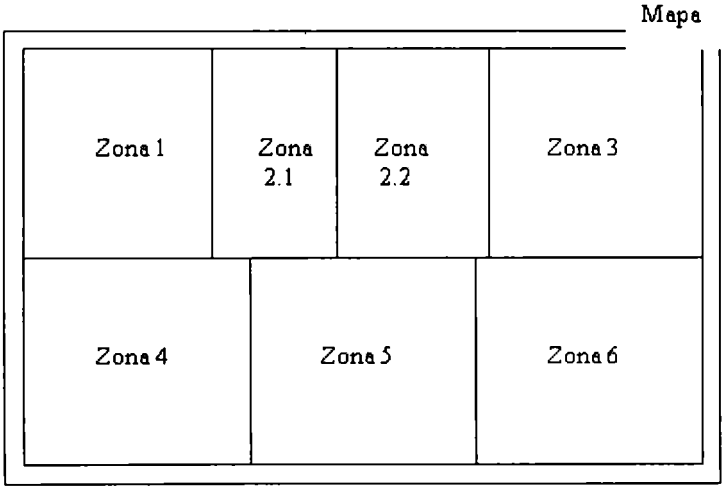


**Figura 6.6: División del mapa en zonas dinámicas teniendo en cuenta el tráfico**

Hasta aquí se observa que una posible estrategia para dividir el mapa en zonas puede estar dada por el tamaño de la zona y la densidad de transito, pero podría ser que esta división no sea buena, por ejemplo, podría ocurrir que en zonas fronterizas la cantidad de sustracciones de vehículos sea mayor, y de esta forma se deseara asignar más patrulleros a esas zonas y estas sean más pequeñas que las céntricas (aún donde la densidad de transito es mayor), de manera que el patrullero pueda cubrir mejor la zona haciéndola más segura. Hasta aquí se tiene que los posibles criterios por los cuales se divide un mapa en zonas son área, densidad de transito vehicular, y cantidad de sustracciones. Pero es posible pensar que la empresa de seguimiento tiene como política ser equitativo con todas las zonas de forma de proveerle a sus afiliados el mismo tipo de seguridad, sin beneficiar solo a algunos, reduciendo su zona o colocando más de un patrullero por zona, con lo cual se puede volver a pensar que el esquema óptimo de división es el de la Figura 6.5.

Después del análisis realizado se puede deducir que la forma de asignar zonas esta muy relacionada con las características del mapa y la decisión que tome la empresa de monitoreo en cuanto a que considerar para realizar la división en zonas.

En la Figura 6.7 se amplia el modelo anterior introduciendo las zonas y estrategias de asignación de zonas.

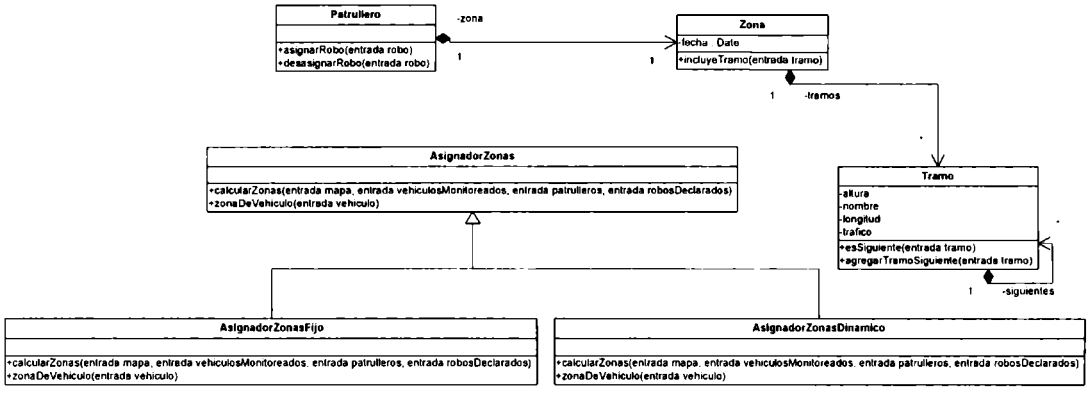


**Figura 6.7: Ampliando el modelo para el cálculo de zonas**

Cuando un patrullero se encuentra en vigilancia tiene asociada la zona que debe patrullar. La zona abarca una extensión del mapa constituida por un conjunto de tramos.

Dada la necesidad de definir zonas por diferentes criterios se define un AsignadorZona, quien se encarga de hacer la distribución de patrulleros y división en zonas. El asignador esta modelado utilizando el patrón de diseñó Strategy [Gamma, 95] que define una jerarquía, donde cada subclase establece una variante del algoritmo para calcular la zona. De esta manera, para definir un nuevo algoritmo para la división en zonas, solo deberá extender la clase AsignadorZonas redefiniendo el método calcularZonas (mapa, vehiculosMonitoreados, patrulleros, robosDeclarados).

Este modelo propone dos subclases de calculador con la siguiente semántica asociada:

- AsignadorZonasFijo: esta subclase redefine el método calcularZonas dividiendo el mapa en zonas fijas de igual tamaño sin tener en cuenta la posición de los patrulleros y vehículos monitoreados.

- AsignadorZonasDinamico: esta subclase redefine el método calcularZonas dividiendo el mapa en zonas de tamaño variable teniendo en cuenta tanto la posición de los patrulleros como la distribución de los vehículos monitoreados, logrando de esta manera tener una buena distribución de vehículos a monitorear por cada patrullero.

### 6.2.5 Introduciendo Algoritmos de Búsqueda de Caminos

Una funcionalidad adicional importante en un sistema de seguimiento vehicular es proveer algoritmos de búsqueda de caminos. Esto es de gran beneficio puesto que ayuda en una persecución de robo, permiten la localización de un camino a un punto de interés tal como un hospital, comisaría, etc.,. Estos caminos debe considerar los diferentes obstáculos que pueden encontrarse en los tramos que componen cada camino.

Dado que puede desearse un camino entre dos puntos teniendo en cuenta diferentes criterios, tales como el más corto, el más económico, el más rápido, etc.; es necesario que el modelo propuesto permita la existencia de diferentes algoritmos de búsqueda de caminos, y que estos puedan ser intercambiados fácilmente entre si sin tener que realizar ningún tipo de modificación en el modelo existente.

La Figura 6.8 muestra las clases participantes para representar tanto los algoritmos de búsqueda como el camino resultante.

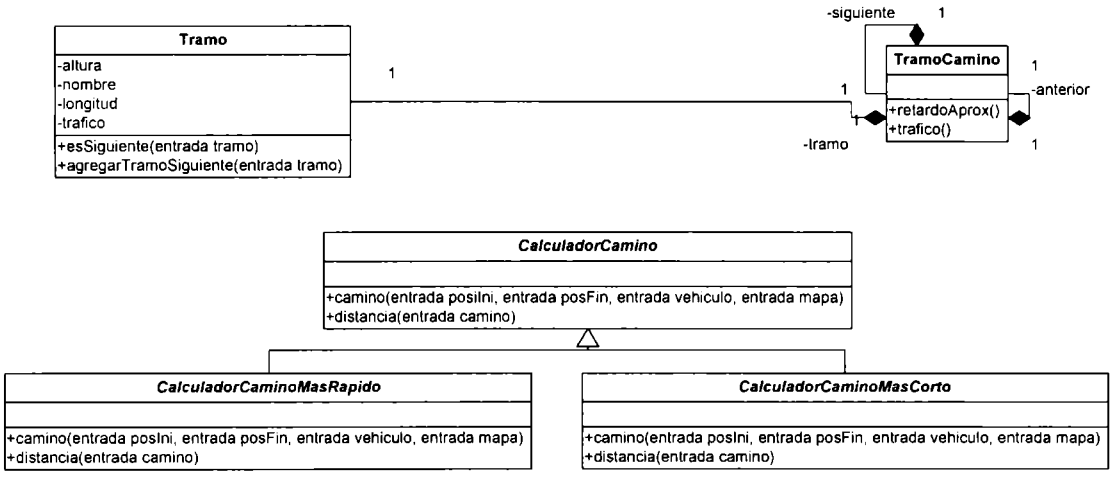


**Figura 6.8: Calculando caminos entre dos puntos.**

Dada la necesidad de una familia de algoritmos intercambiables para la búsqueda de caminos, donde cada uno provee una variante para dicha búsqueda, se hace uso nuevamente del patrón de diseño Strategy [Gamma, 95] el cual permite realizar esto sencillamente.

La clase abstracta CalculadorCamino representa la estrategia necesaria y es quien proporciona el protocolo común para determinar el camino y la distancia del mismo. El protocolo o interface de esta clase es: camino (posicionInicio, posicionFin, vehiculo, mapa) y distancia(camino). Cada estrategia concreta será una subclase de esta clase abstracta quien implementará dicha interface.

El resultado del método camino, debe ser una instancia de la clase TramoCamino, la cual representa el punto inicial del mismo, quien conoce al siguiente

tramo que lo compone y siguiendo esta secuencia se llega al último tramo del camino, quien es el destino final.

### 6.2.6 Asignando Patrulleros a un robo

Cuando un vehículo es robado un conjunto de patrulleros debe encargarse de perseguir el vehículo con el objetivo de lograr su recuperación. En el momento que el sistema es notificado de la ocurrencia de un robo, un conjunto de patrulleros debe asignarse al la persecución del vehiculño sustraido. La forma en que se asigna patrulleros a la persecución de robos puede depender de varios criterios, y podrían existir diferentes estrategias de asignación. Algunas de ellas pueden ser: asignar los 5 patrulleros más cercanos al vehículo robado, asignar un solo patrullero al robo, asignar n vehículos al robo donde el n es tomado en función de la cantidad de patrulleros que se encuentran en vigilancia, etc.

La Figura 6.9 muestra un modelo que permite la asignación de patrulleros a un robo y admite el intercambio de estrategias independientemente del cliente que lo use.

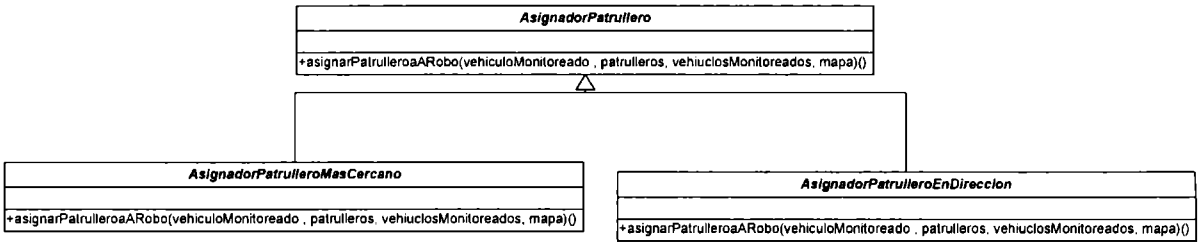


**Figura 6.9: Asignando patrulleros a un robo.**

Debido a que existen diferentes criterios para determinar cuales y cuantos patrulleros asignar a un robo, se ha decidido aplicar nuevamente el patrón de diseño Strategy [Gamma, 95] para determinar los mismos. Subclasificando la clase abstracta AsignadorPatrullero e implementando el método asignarPatrulleroARobo(vehiculoRobado, patrulleros, vehiculosMonitoreados, mapa) se pueden crear nuevas estrategias de asignación. En este diseño se muestran dos implementaciones de asignadores de patrulleros, "AsignadorPatrulleroMasCercano" el cual asigna los tres patrulleros más cercanos a la posición del vehículo robado que se encuentren en estado de vigilancia y "AsignadorPatrulleroCercado" el cual asigna los 4 patrulleros que se encuentran más cercanos y logren rodear al vehiculo evitando su huida en cualquier direccion. En la Figura 6.10 se muestra cual sería la asignación del "AsignadorPatrulleroCercado".

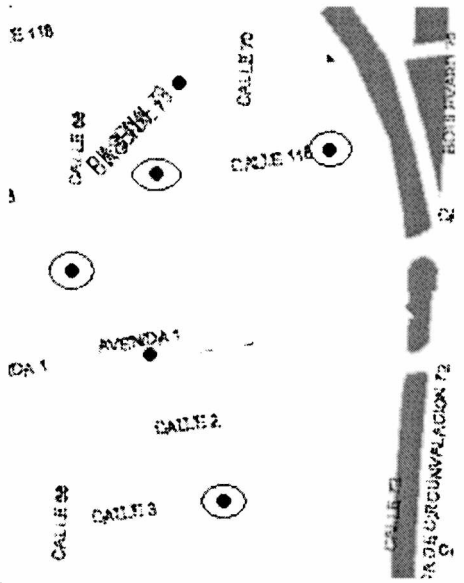


**Figura 6.10: Ejemplo de asignación de patrullero por cercado**

En la figura 6.10 se señala con un punto rojo un vehículo que ha sido robado y con puntos azules los patrulleros, encontrandose destacados con un círculo verde aquellos que serían asignados al robo a través de la estrategia AsignadorPatrulleroCercado.

Los parámetros que recibe asignarPatrulleroARobo prevén que alguna estrategia podría tener en cuenta no solo el vehículo robado y la posición y estado de los patrulleros, sino también la situación del resto de los vehículos monitoreados, para determinar la cantidad de patrulleros asignados y no dejar desprotegidos al resto de los vehículos monitoreados o zonas del mapa.

### 6.2.7 Incluyendo el Objeto AVL

Hasta el momento solo se han mostrado los diferentes componentes necesarios para un sistema AVL pero no se ha expresado como ellos son conocidos y establecidos para que el sistema pueda interactuar con ellos. Es aquí donde se observa la necesidad de una entidad que englobe las diferentes componentes del sistema AVL y a través de la misma se pueda determinar cada parte que la compone e interactuar con ella. Esta entidad u objeto representa el AVL utilizado por la aseguradora y debe conocer los patrulleros disponibles, los vehículos monitoreados por la empresa, el mapa de la ciudad o ruta que se esta vigilando, los puntos de interés encontrados en dicho mapa, los robos declarados y recuperaciones realizadas por la empresa, además de las estrategias que utilizara para asignar patrulleros a robos, asignar zonas, y calcular los recorridos entre dos puntos.

La Figura 6.11 muestra el objeto AVL junto con su relación con las clases directas.

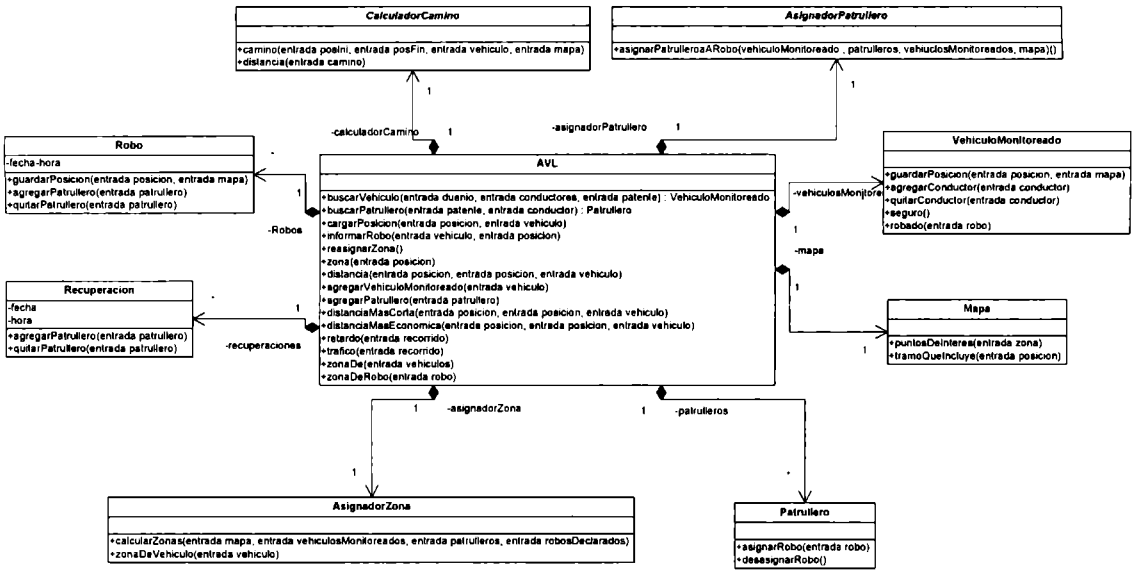


**Figura 6.11: El corazón del sistema de localización automática de vehículos.**

Además de conocer las clases representativas de cada uno de los componentes necesarios por el sistema AVL con los cuales necesita interactuar, la clase AVL, quien representa el sistema de localización automática de vehículos utilizado por la aseguradora, posee una interfaz que establece el comportamiento de esta clase y el sistema en general, dado que esta clase es el punto de entrada a todo el sistema AVL detallado.

## *6.3 Modelado de contextos*

### 6.3.1 Introducción

Para trabajar con una aplicación sensible al contexto es necesario realizar un modelo que represente la información contextual. El significado de los términos contexto y sensibilidad al contexto fue desarrollado previamente en el capítulo 4, por lo que aquí solo se tratan las diferentes formas de modelado y utilización de esta información.

En primer lugar se hará un relevamiento de trabajos relacionados al modelado de contexto en el cual se incluye la captura y adaptación al mismo, para finalmente presentar el modelo propuesto y utilizado en este y trabajo.

### 6.3.2 Trabajos relacionados

En las sub-secciones siguientes serán tratados tres diferentes enfoques en relación al tema: modelado del contexto, captura de la información contextual del ambiente y adaptación de la aplicación al contexto o sensibilidad al contexto.

### 6.3.2.1 Aproximaciones de modelado

Existe un amplio número de formas de representar el contexto, estas van desde colecciones de datos primitivos hasta complejas arquitecturas de objetos. En esta sección se presenta un relevamiento de aproximaciones de modelado conocidas.

#### *6.3.2.1.1 Key-value models*

Es el modelo más simple de estructurar los datos para modelar la información contextual. Es fácil de manejar, pero existe una perdida de sofisticación en su estructura de forma que dificulta algunos algoritmos de recuperación de información.

Las colecciones de datos primitivos son flexibles pero, dependiendo los requerimientos del sistema, pueden no ser eficientes para representar la realidad deseada. Otra desventaja es que el costo de la representación simple lleva asociada una codificación y decodificación para su interpretación.

La ventaja de esta representación simple es que puede ser utilizada en cualquier aplicación. Además, con esta representación los costos de envió a través de la red son lo más reducidos posibles.

El modelo de key-value para modelar contexto fue usado por Schilit et al. [Schilit, 94] proveyendo el valor de la información de contexto como una variable de entorno.

#### *6.3.2.1.2 Markup Scheme Models*

Estos esquemas son estructuras de datos jerárquicas consistentes de tags (etiquetas) con atributos y contenido. El contenido de los tags es definido por otros tags. Existen un gran número aproximaciones de modelado con marcas, entre ellos CC/PP (Composite Capabilities / Preferences Profile) [Indulska, 03], UaProf (User Agent profile) [UaProf], CSCP (Comprehensive Structured Context Profiles) [Held, 02] , Profile Description Language (PPDL)[Chtcherbina, 03].

La Figura 6.12 introduce un ejemplo de modelado de contexto a través de marcas.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cscp="context-aware.org/CSCP/CSCPProfileSyntax#"
  xmlns:dev="context-aware.org/CSCP/DeviceProfileSyntax#"
  xmlns:net="context-aware.org/CSCP/NetworkProfileSyntax#"
  xmlns="context-aware.org/CSCP/SessionProfileSyntax#"
  <SessionProfile rdf:ID="Session">
    <cscp:defaults rdf:resource=
      "http://localContext/CSCPProfile/previous#Session"/>
    <device><dev:DeviceProfile>
      <dev:hardware><dev:Hardware>
        <dev:memory>9216</dev:memory>
      </dev:Hardware></dev:hardware></dev:DeviceProfile>
    </device>
  </SessionProfile>
</rdf:RDF>
```

**Figura 6.12: Ejemplo de CCPP**

#### *6.3.2.1.3 Graphical Models*

Un instrumento de modelado muy general también usado aquí es UML, el cual debido a su genericidad es apropiado para modelar los contextos. Un ejemplo de su uso se encuentra en el trabajo de Bauer [Bauer, 03], donde aspectos contextuales a una aplicación de administración de vuelos es modelado como extensión de UML.

#### *6.3.2.1.4 Object-Oriented Models*

En la aproximación de modelado orientado a objetos la intención en el modelado de contextos es emplear los principales beneficios del paradigma. Los detalles del procesamiento de contextos son encapsulados en objetos. El acceso a la información contextual es provisto a través de su interfaz. Un trabajo representativo de esta aproximación es el realizado por Cheverst et al. [Cheverst, 99], el cual realiza un modelo de objetos para una aplicación turística.

#### *6.3.2.1.5 Logic Based Models*

La lógica define las condiciones en las cuales expresiones o hechos pueden ser derivados desde un conjunto de otras expresiones o hechos. Para describir estas condiciones en un conjunto de reglas un sistema formal es aplicado. Consecuentemente en un sistema lógico, los contextos son definidos como hechos, expresiones y reglas. Usualmente, información contextual es agregada, actualizada o borrada de un sistema lógico, o es inferida desde reglas existentes en el sistema. Como todos los sistemas lógicos su modelado requiere un alto grado de formalidad. Una de las primeras aproximaciones ha sido publicada en Formalizing Context [McCarthy, 93].

#### *6.3.2.1.6 Ontology Based Models*

Las ontologías son instrumentos para especificar conceptos e inter-relaciones y están particularmente disponibles en partes de proyectos.

A su vez, proveen una manera uniforme para especificar los conceptos fundamentales de modelos, así como también una cantidad arbitraria de subconceptos y hechos. Esto permite compartir y reusar información contextual en sistemas ubicuos.

Una de las primeras aproximaciones para modelar contextos con ontologías fue propuesto por Otztürk et al.[Otztürk, 97]. Ellos analizan estudios psicológicos en la diferencia entre el recuerdo y el reconocimiento de varios problemas en relación con la información contextual. Desde este punto de vista estos derivan la necesidad de formalizar y combinar el conocimiento desde diferentes dominios.

### 6.3.2.2 Arquitecturas para capturar contextos

A continuación son detalladas arquitecturas que permitan capturar el concepto de contexto. Estas abarcan la captura del contexto, pero todas ignoran la fase de adaptación al mismo.

El objetivo principal planteado es que quien programe una aplicación debe encontrarse pendiente del contexto, y no del proceso de cómo capturarlo.

### *6.3.2.2.1 Context Toolkit Approach*

El Context Toolkit Approach [Dey, 99] apunta a separar la adquisición de contexto del envió y uso del mismo. El Context Toolkit soporta la adquisición y envió de contexto usando tres tipos de abstracciones: widget, servers e interpreters.

Context widgets son componentes de software que proveen a aplicaciones acceso al contexto censado desde el ambiente operativo. Ellos liberan a las aplicaciones del proceso de adquisición de contexto ocultando la complejidad de la toma de datos desde el sensor a las aplicaciones. Cada widget encapsula el estado y un conjunto de eventos de callback (llamada). El estado abarca información contextual que las aplicaciones pueden explotar vía polling o subscripciones. Callback representa los tipos de eventos que los widget pueden usar para notificar a las aplicaciones subscriptas. Los widget también mantienen el estado contextual permitiendo a otros componentes retornar información de contexto histórica.

Context servers son usados para recolectar el contexto sobre una entidad particular, tal como una persona. El servidor de contexto es responsable de suscribir a cada widget de interés y actúa como un proxy a la aplicación, recolectando información para una entidad particular. El servidor de contexto puede ser visto como un widget compuesto y, como tal, tiene atributos y callbacks, pueden efectuarse subscriciones y polling a él, y retornar información histórica.

Context interpreters son responsables de implementar la interpretación de información de contexto. Transforman los diferentes formatos de representaciones o mezclan diferentes informaciones de contexto para proveer nuevas representaciones.

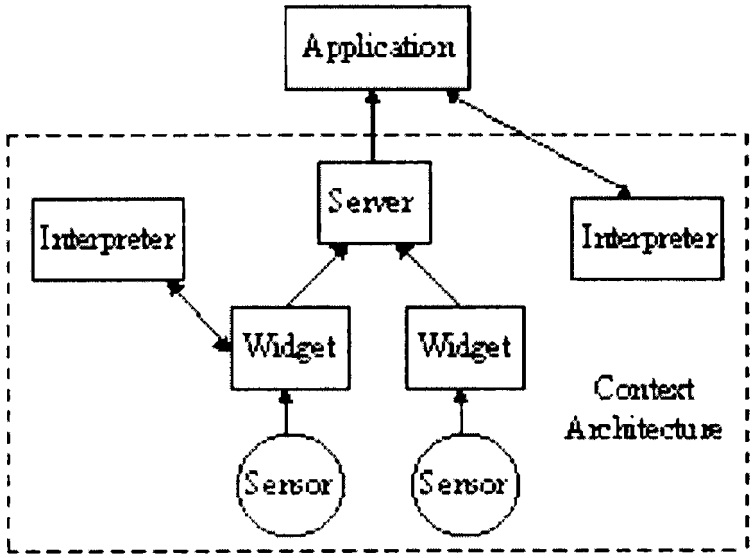


**Figura 6.13: La arquitectura del Context Toolkit**

Con este modelo, los Context Widget manejan información puntual, la cual puede ser interpretada y representada por Context Servers. Las aplicaciones que reciben notificaciones sobre un tipo particular de contexto, como la actividad con una ubicación específica, pueden subscribirse para eventos desde el widget real.

No obstante, si eventos sobre entidades de un sistema particular tal como usuarios son requeridos, entonces la aplicación debe registrar las notificaciones al Context Server. Aunque el Context Toolkit incluye mecanismos para habilitar

aplicaciones a registrar callbacks, esto no es inherente para soportar movilidad del usuario debido a las posibles desconexiones desde la red ocacionadas por ésta.

#### *6.3.2.2.2 The Hydrogen approach*

Schwinger et.al [Schwinger, 02] proponen una arquitectura de 3 capas para separar la problemática de la interacción con los sensores físicos, el almacenamiento y mantenimiento del contexto de la aplicación en si mismo. Las capas que maneja son el Adaptor Layer, el Management Layer, y el Application Layer.

El componente central es el almacenamiento para el contexto, el cual puede ser consultado por aplicaciones, llamado Context Server. Los contextos son almacenados una vez por cada dispositivo y son provistos a todas las aplicaciones.

La Figura 6.14 muestra la arquitectura utilizada por hydrogyn.

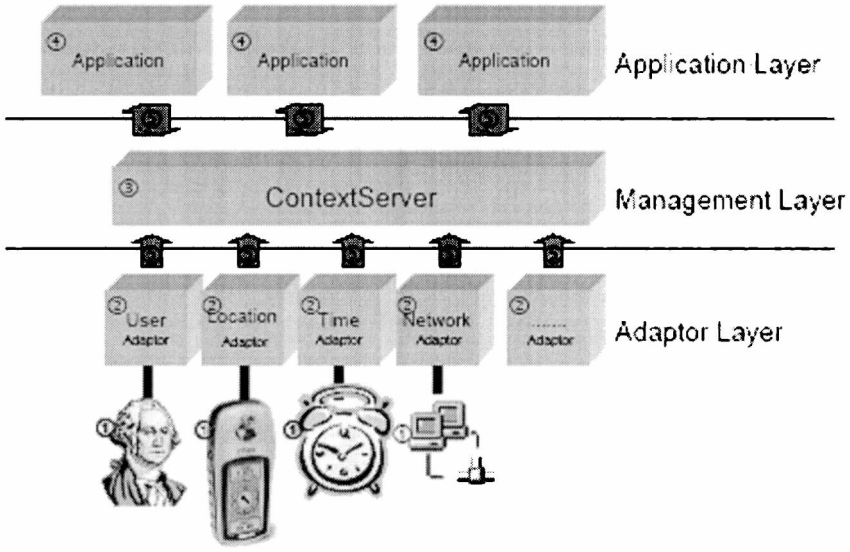


**Figura 6.14: Arquitectuta hydrogyn**

El Adaptor layer es responsable de tomar información desde sensores sobre el contexto físico, posiblemente enriquece esta información con información de contexto lógico y la envía al Management Layer. Adicionalmente el ContextServer embebido en el Management Layer provee métodos simples para que la aplicación pueda subscribirse a contextos y obtener información de estos. Además, problemas asociados con los dispositivos que maneja cada aplicación son simplificados, ya que los adaptadores permiten ser intercambiados.

Por último, el Application Layer representa las aplicaciones que usan el contexto proporcionado por las capas inferiores.

El hydrogyn approach plantea que las razones para separar el censado de contexto del almacenamiento es que dos aplicaciones pueden tener acceso simultáneo a los contextos.

#### *6.3.2.2.3 El Blackboards (Winograd)*

La arquitectura de Blackboards propuesta por Winograd [Winograd, 01] adopta datos centralizados más que un punto de vista centralizado. Aunque se envíen requerimientos a componentes distribuidos y se tomen callbacks desde ellos, se efectúa

un proceso de envió de mensajes a un pizarrón compartido y común, y pueden suscribirse para recibir mensajes que matcheen con un patrón específico.

En una arquitectura de blackboards, todas las comunicaciones van a un servidor centralizado. El ruteo a diferentes componentes es efectivamente realizado por el matching de mensajes contenido en el patrón de suscripción. Cualquier cosa que puede ser hecha con comunicación directa a un lugar, puede ser simulada incluyendo el identificador del camino, como un campo en el mensaje y usando matching para ver si es el componente deseado.

#### 6.3.2.3 Mecanismos de Adaptación a contextos

Existen diferentes mecanismos utilizados para la adaptación de contextos, uno de ellos es el propuesto por el proyecto UWA (Ubiquitous Web Applications) que utiliza un mecanismo de adaptación basado en reglas. A continuación será detallada la propuesta realizada en este proyecto, no obstante, para un mayor entendimiento del mismo será también mostrado el modelo de contexto presentado por dicho proyecto.

##### *6.3.2.3.1 Diseño del modelo UWA*

UWA (Ubiquitous Web Applications) Kappel et al. [Kappel, 02] proporciona una representación para el modelo de contexto y un modelo de reglas para manejar la adaptación al contexto.

El modelo implementado por UWA fue pensado para aplicaciones web ubicuas pero los conceptos desarrollados en éste pueden ser aplicados a cualquier aplicación ubicua.

Dicho modelo comprende:

- Un modelo de contexto
- Un modelo de reglas para adaptación

El modelo de contextos manejado por UWA divide el contexto en dos representaciones, una física, que representa el nivel de contexto representado por los sensores de ambiente, y una lógica, que representa una abstracción de la información recolectada. Un mecanismo basado en reglas, en términos de reglas de adaptación, es empleado para especificar las adaptaciones. Las reglas de adaptación, a su vez, son determinadas por los requerimientos.

**UWA** propone un modelo de adaptación genérico en el sentido de un framework orientado a objetos, el cual provee al diseñador de la adaptación con elementos del modelo apropiado. Que sea genérico significa que el modelo de la aplicación es independiente y provee algunas clases predefinidas y lenguajes para construir aplicaciones adaptables. Además, las clases predefinidas pueden ser extendidas por el diseñador de la adaptación a través de subclasificación, para los detalles específicos de la aplicación. Los modelos genéricos de adaptación comprenden modelos de contextos junto con un modelo de reglas de adaptación y varios submodelos.

## *Modelando contexto*

En el modelo el censado de la información contextual y la actualización esta fuera del alcance del mismo. En esta aproximación se asume que cada vez que los valores cambian son actualizados automáticamente.

El modelado se basa en la definición de contexto como "abstracción de ciertas propiedades, describiendo el entorno de la aplicación y algunos aspectos de la aplicación en si misma" necesarios para determinar la necesidad de adaptación.

De acuerdo al nivel de abstracción se definen dos modelos de contexto: físico y lógico.

### Modelo de contexto Físico

Las propiedades del contexto físico son abstracciones de bajo nivel y son actualizadas dinámicamente para tomar en cuenta que el ambiente y los estados de las aplicaciones cambian continuamente.

Aunque la aplicación de contexto es independiente de la aplicación, el diseñador de la aplicación debe especificar cuales de las propiedades del contexto utilizará.

El diagrama UML muestra el modelo de contexto físico usado por UWA project. donde el contexto físico es una agregación de las propiedades aquí descriptas. El modelo puede ser extendido agregando clases de propiedades de contexto físico adicionales por subclasificación de PhisicalContextProperty.

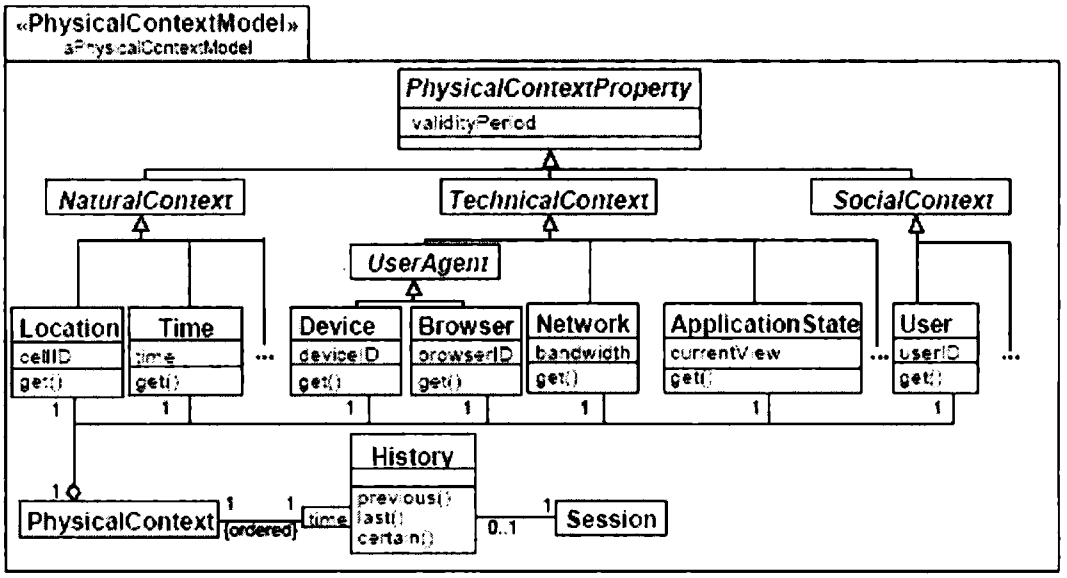


**Figura 6.15: Modelo de contexto físico de UWA Project**

### Modelo de contexto lógico

En contraste al modelo de contexto físico, el modelo de contexto lógico representa información más abstracta del contexto. Básicamente, la información de contexto lógico es necesaria para enriquecer la semántica de la información de contexto físico dándole significado para la posterior adaptación. Así, para cada propiedad que es parte del modelo de contexto físico, el modelo de contexto lógico provee información

lógica apropiada en término de los llamados perfiles. De forma similar al contexto físico el modelo de contexto lógico puede ser fácilmente extendido por el diseñador para representar clases adicionales de perfiles necesarios para una aplicación particular.

Cada uno de las propiedades de contexto física tiene una representación de perfil lógico. Debido a esto existen los siguientes perfiles

- Ubicación (Location Profile): provee una variedad de información genérica sobre la cartografía física y política, con la posibilidad de especificar características propias de aplicaciones específicas.
- Tiempo (Time Profile): provee información genérica para aplicaciones específicas sobre el tiempo, dando una interpretación al contexto físico.
- Dispositivo (User Agent Profile): captura las propiedades del dispositivo en términos de software y hardware de forma independiente. En lo que es hardware se tiene en cuenta información de las capacidades del dispositivo como por ejemplo tamaño del dispositivo, color y gráficos soportados, memoria, procesador y capacidad de conexiones. Con respecto a software maneja información sobre que sistema operativo, browser, etc. son utilizados.
- Red (Network Porfile): provee información genérica sobre las conexiones de red disponibles, tal como GMS, HSCSD y GPRS.
- Usuario (User profile): comprende información que es voluntariamente ingresada por el usuario, describiendo las preferencias del usuario con respecto a la adaptación, así como también la información que es transparentemente adquirida por el sistema.

### *Mecanismo de adaptación*

Además del modelado de contexto, UWA propone un mecanismo de reglas de adaptación llamado ECA (Event/Condition/Action)

**Event:** El evento determina los eventos que dispararan una regla.

**Condition:** La condición es evaluada cuando la regla es disparada por un evento. Ésta chequea cuando una adaptación es requerida y si esta es apropiada.

**Action:** La acción es responsable de ejecutar una adaptación de la aplicación.

### Modelo de eventos

Los eventos de este modelo necesitan monitorear cambios en la información de contexto física y lógica. El modelo de eventos especifica ciertos eventos predefinidos que pueden ser adaptados por el diseñador de las reglas. Los eventos pueden ser tanto primitivos como compuestos. Los primitivos indican cambios tanto en el contexto lógico como el físico, y los compuestos relacionan eventos para representar situaciones del mundo real.

La Figura 6.16 muestra el diagrama UML del modelo de eventos

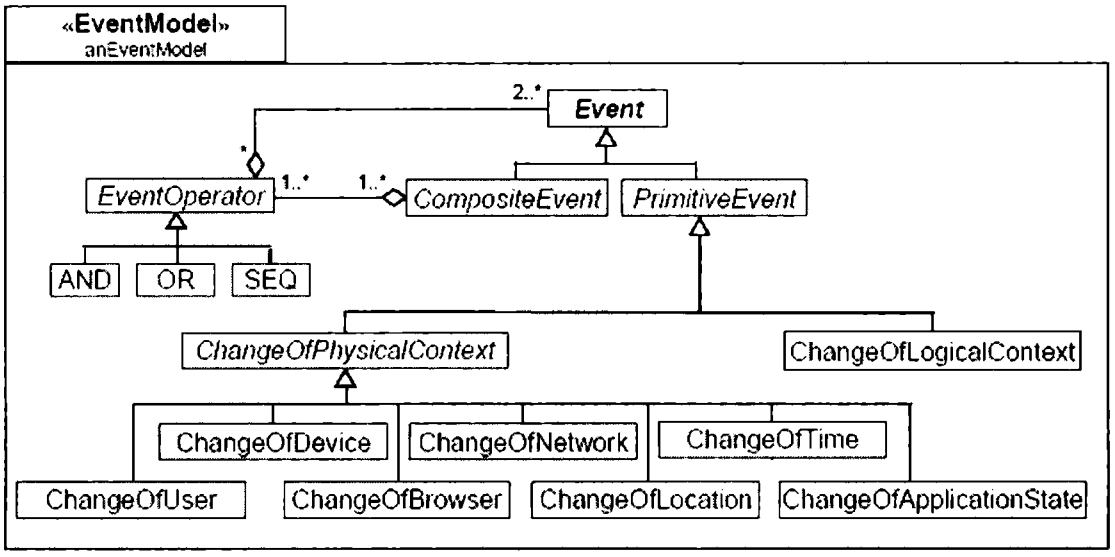


**Figura 6.16: Modelo de eventos de UWA**

### Modelo de condiciones

En general, las condiciones pueden ser formuladas directamente con reglas ECA (o parte de ellas). UWA propone utilizar OCL (Object Constraint Languaje) [Rumbough, 98] para especificar condiciones.

### Modelo de acciones

Las adaptaciones requeridas son realizadas dentro de las acciones en el modelo de reglas ECA. El modelo de acciones genérico mostrado en la Figura 6.17 muestra una especificación de las acciones que realizarán las operaciones de adaptación. Básicamente, la parte de acciones puede tener una lista de acciones a ser ejecutadas (ActionList), compuesta de al menos una acción (Action). Una acción puede ser una acción atómica (AtomicAction) o un bloque de acciones (BlockAction). Las acciones atómicas pueden ser la asignación de un valor (Assigment) o la invocación de una operación de adaptación. A su vez, esta última, puede ser la invocación de una operación de adaptación específica de la aplicación (ApplicationSpecificAO) o la invocación de una operación de adaptación genérica (GenericAO).

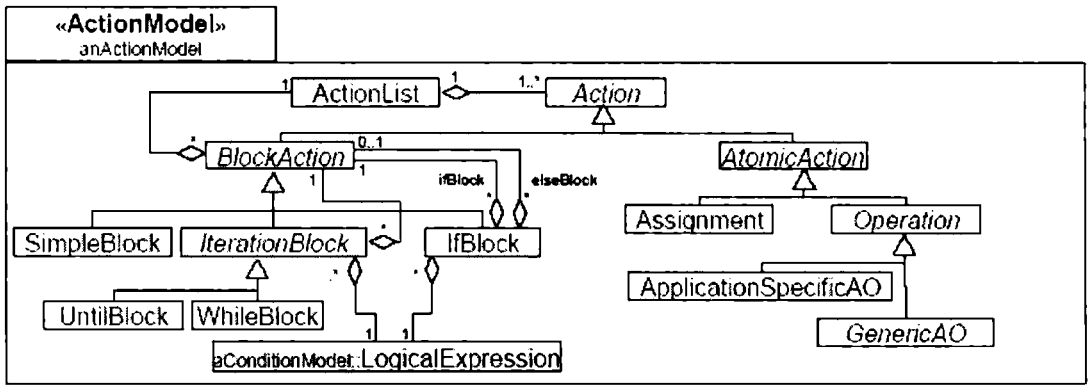


**Figura 6.17: Modelo de acciones de ECA**

Un bloque de acciones puede ser un bloque simple (SimpleBlock), un bloque condicional (IfBlock) o un bloque iterativo (IterationBlock). Excepto el bloque simple,

los bloques tienen asociada una condición lógica que determina si el bloque se debe ejecutar o la condición de terminación de la iteración.

### 6.3.3 Modelo utilizado

Como fue expuesto en las secciones anteriores existen varias aproximaciones para el modelado del contexto y diferentes enfoques dentro del mismo.

En lo que respecta a una aproximación de modelado del contexto, esta tesis propone una aproximación orientada a objetos, debido a las ventajas ya conocidas del paradigma.

En base a la literatura existente y considerando las limitaciones de la tecnologías actuales, se propone un modelado de contextos físicos y lógicos embebidos en la aplicación.

#### 6.3.3.1 Modelo de contexto físico

Para el modelado de contextos físicos se toma una posición cercana a la de UWA, en el sentido de ver al contexto como un conjunto de propiedades. No obstante, se decide no hacer una jerarquía en base a cada una de las propiedades de contexto existentes, como es presentado en la Figura 6.15, dado que el contexto físico solo intenta capturar información y esta no tiene ni comportamiento ni información común con otras propiedades. Por estos motivos la subclasificación en diferentes tipos de propiedades de contexto (context properties) no se consideró apropiado. Las propiedades de contexto son datos simples, donde su único objetivo es guardar un valor para cada una de las propiedades tenidas en cuenta en el contexto. A su vez, cada propiedad de contexto puede tener asociado un conjunto de atributos. Por ejemplo, dada la propiedad de contexto de dispositivo se podrían tener en cuenta si su pantalla es a color, ancho y alto de la misma, memoria del dispositivo, etc.

Dado que podrían agregarse nuevas propiedades de contexto, o nuevos atributos a dichas propiedades a medida que sea requerido, fue provechosa la utilización del patrón de diseño "type object", el cual desacopla las instancias de sus clases implementando dichas clases como instancias de una clase. Type object permite que nuevas clases sean creadas en tiempo de ejecución [Johnson, 97].

La Figura 6.18 muestra el diseño adoptado para la representación del contexto físico.

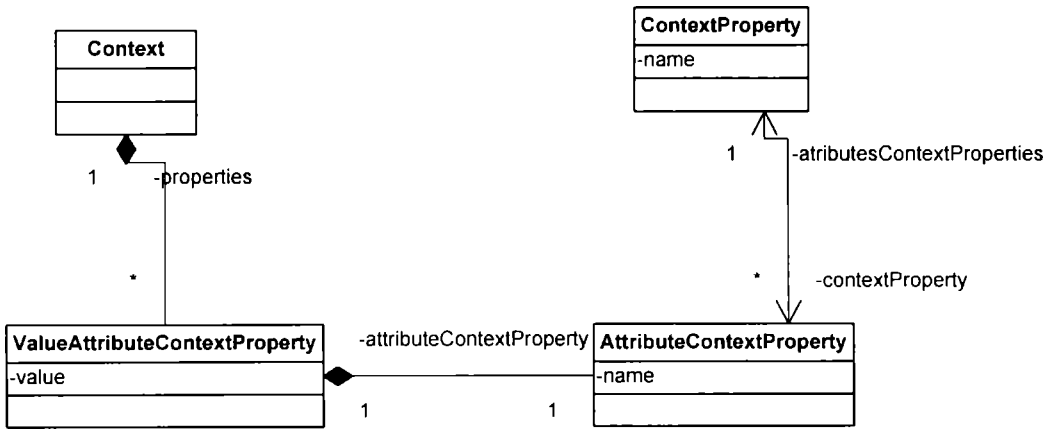


**Figura 6.18: Modelo de contexto físico**

En la figura puede observarse el uso del patrón type object en dos oportunidades, una entre las clases ContextProperty, quien representa una propiedad de contexto (por ejemplo, la propiedad de dispositivo), y AttributeContextProperty, que representa un atributo de la propiedad de contexto (por ejemplo, colores de pantalla del dispositivo). La segunda relación se establece entre las clases AttributeContextProperty y ValueAttributeContextProperty, quien establece un valor para dicha atributo (por ejemplo, el valor 256 para el atributo colores de pantalla).

Por otro lado, ContextProperty conoce cuales son sus atributos, lo cual permite evitar la creación de uno nuevo que represente el mismo atributo para la propiedad.

Por último, un contexto conoce todos los valores de propiedad que son tomados en cuenta para el mismo, pudiendo conocer a través de ellos de que atributo y de que propiedad se trata.

En la aplicación la información de contexto viaja entre el usuario móvil y la central. Esta representación del contexto físico minimiza la cantidad de información que viajará a través de la red, ya que si una propiedad de contexto no es tenida en cuenta en algún momento o no se tiene un valor de ella (esto puede ocurrir, por ejemplo, porque no se disponga de un sensor para la capturara de algún dato) puede optarse por no añadirle valores de atributos de la misma.

#### 6.3.3.2 Modelo de contexto lógico

En cuanto al modelo de contexto lógico, gran parte de la información contextual sobre los usuarios móviles se encuentra embebida en nuestro modelo de base de la aplicación AVL. Es así, por ejemplo, que un patrullero conoce cual es su ubicación actual y por donde se movió a través de su recorrido en dicho modelo, manejando de esta manera información lógica sobre su contexto de ubicación, siendo la ubicación (Location) modelada en el modelo de base. Otra característica tenida en cuenta en el modelo es que un patrullero conoce información sobre su estado (Persecución o Vigilancia), y se posee información sobre quien es el conductor.

En base a esto puede observarse que manejar información lógica fuera de este modelo seria ineficiente e improductivo para la aplicación, dado que esta ya se encuentra incluida en el modelo AVL.

Para algunas propiedades de contexto, tal como dispositivo y red, solo es necesario conocer algunos valores de atributos de las mismas para realizar adaptaciones en respuesta a solicitudes de usuarios móviles, es por esto que no se respeta completamente lo planteado por UWA, quien establece que cada propiedad de contexto físico tiene su equivalente representación lógica. Para esta tesis no se cree necesario dicho proceso de abstracción ya que solo son necesarias propiedades simples para la adaptación, por ejemplo, no se toma en cuenta la red sino algún atributo como ancho de banda y se adapta en base a este atributo. Por estos motivos, el modelado de contexto físico para estas propiedades es suficiente.

#### 6.3.3.3 Captura de datos desde el ambiente

En cuanto a las técnicas de captura de datos esta tesis no se detiene en el detalle de cómo obtener datos desde el ambiente, se asume que ciertos datos son obtenidos sin preocuparse como es que son cargados en el contexto físico.

En contraposición se tiene en cuenta la ocurrencia de eventos que disparan cambios en el contexto, como por ejemplo la ocurrencia de un robo que modifica el estado de un patrullero cambiando su estado a persecución. La captura de estos eventos es llevada a cabo a través de observadores como se verá en la sección 6.4.5.

#### 6.3.3.4 Adaptación al contexto

En lo que respecta al mecanismo de adaptación de la aplicación móvil en base al contexto del usuario móvil, fueron encontrados dos casos. En el primero se detecto que cierta información contextual ya se encontraba embebida en el modelo AVL para realizar dichas adaptaciones, como es el caso de la asignación de zonas para los patrulleros pertenecientes al sistema, el cual varía dependiendo del estado del patrullero (persecución o vigilancia) y su ubicación. Dicha adaptación surge naturalmente del modelo sin ser necesario otros mecanismos externos al mismo para su realización.

Un segundo caso ocurre cuando las adaptaciones al contexto no surgen naturalmente del modelo, como pueden ser adaptaciones a un dispositivo o características de la red de comunicaciones, que no son propios a un modelo de base. Un ejemplo de esto puede ser la adaptación a la pantalla del dispositivo del usuario, donde dependiendo de esta será la calidad del mapa que se le suministrará (teniendo en cuenta la resolución) y la cantidad de colores que posea (dependiendo de la calidad de colores del dispositivo).

Este último tipo de adaptación es realizada cuando un usuario móvil hace un requerimiento o cuando se le notifica un evento, dándole la respuesta adecuada. Estas adaptaciones podrían ser realizadas utilizando el mecanismo de reglas de adaptación ECA propuesto por UWA.

## *6.4 Infraestructura de comunicaciones*

Entre las tareas que se debieron realizar se incluyo la investigación del funcionamiento de las redes inalámbricas y la estructura que éstas utilizan para diseminar la información en dispositivos móviles. Luego de obtener esta información las comunicaciones fueron desarrolladas teniendo en cuenta las siguientes características:

**Escalabilidad:** el sistema debe ser capaz de soportar un potencial número de usuarios que requieren acceso simultáneo.

**Flexibilidad**: el sistema debe soportar tanto requerimientos de servicios como el envío de información a los usuarios móviles para notificarles eventos o suministrarle información sin que ellos hayan disparado un requerimiento.

**Soporte para desconexiones**: Dado que no es posible asegurar la existencia de conexión a la red en todo lugar de la ciudad y en todo momento, el sistema debe ser capaz de sobrevivir a períodos de desconexión.

### 6.4.1 Esquema de comunicaciones

En el sistema se consideran dos esquemas diferentes de comunicaciones que dependen del tipo de vehículo con el que se esta trabajando.

Los vehículos monitoreados toman su posición exacta en un instante de tiempo a través de un sensor de posicionamiento y transmiten dicha posición a través de un dispositivo de comunicación. A su vez, pueden emitir una señal de robo del vehículo, la cual puede ser activada desde un dispositivo colocado en el mismo. En este sentido la comunicación se estable unidireccionalmente desde el vehículo monitoreado hacia una central receptora de información. La información que se envía es simplemente la posición y un identificador del vehículo y, en el caso de desear informar un robo, solo viajara la señal y un identificador del vehículo substraído.

En el caso de los patrulleros la comunicación se da bidireccionalmente dado que los patrulleros establecen una comunicación con su central para solicitar un servicio o enviar información de su contexto y, la central entabla comunicación con los patrulleros para informarles de ciertos eventos tal como la asignación a persecución de robo de un vehículo.

Los eventos por los cuales la central comienza una comunicación con el patrullero son:

- Asignación de persecución a un robo.
- Desasignación de persecución a un robo.

Otros eventos, tal como el cambio de ubicación del resto de los vehículos, no son enviados como información por la central sino que el usuario móvil es el encargado de solicitar un servicio de actualización de los datos.

### 6.4.2 Modelo de comunicaciones

Desde un punto de vista abstracto las comunicaciones realizadas en el sistema pueden verse como usuarios móviles que se conectan y hacen requerimientos de manera similar a la que lo harían a un servidor web, utilizando una red inalámbrica mediante el protocolo HTTP. Este protocolo especifica básicamente el modo de comunicación entre un cliente y un servidor. El cliente solicita un documento, imagen o archivo del espacio de direcciones del servidor, y éste se lo provee. La arquitectura cliente/servidor se basa en el paso de mensajes de un dispositivo o máquina (cliente) que realiza las peticiones de servicio a otra en la que residen los datos y programas de aplicación (servidor). Entre las ventajas de esta arquitectura se encuentran la usabilidad, flexibilidad, interoperabilidad y escalabilidad.

Dado que a su vez los usuarios móviles (o patrulleros) también deben recibir información desde el servidor, se decidió hacer una capa que maneje las comunicaciones entre ambos.

La comunicación entre el usuario móvil (o patrullero) y la central, que posee la información, básicamente cae en alguno de los siguientes puntos:

- Logueo al sistema y validación de usuario

  Aquí es donde el usuario debe ingresar una identificación del patrullero de la empresa en el que se encuentra (su patente) y su clave personal, estos datos son

enviados vía un requerimiento HTTP al servidor, el servidor valida la existencia de dicho patrullero (a través de la patente) y la coincidencia de la clave personal con la de las personas autorizadas a conducirlo..Una vez que el usuario ha sido validado puede comenzar a hacer requerimientos de información.

- Preparación para la recepción de eventos desde la central

  En este punto, el patrullero debe prepararse para recibir cualquier evento que envié la central pero a su vez debe continuar con su ejecución normal para poder realizar otras operaciones en paralelo. Es por esto que se necesita desprender un hilo de ejecución que espere por la comunicación de cualquier evento y en caso de recepción actúe de manera adecuada.

- Solicitud de requerimientos

  Luego del logueo y estando preparado para la recepción de eventos, el usuario móvil puede comenzar a hacer requerimientos, los cuales se harán vía requerimientos HTTP al servidor

- Recepción de eventos

  Cuando la aplicación móvil recibe un evento enviado desde la central, el cual es recepcionado por el hilo de ejecución encargado de esa tarea, debe realizar las operaciones pertinentes al evento ocuarrido.

- Salida del sistema

  Una vez que el usuario ha decidido abandonar la aplicación móvil debe comunicárselo a la central para que ya no envíe los eventos asociados al patrullero, cierre dicha comunicación y no lo considere activo en el sistema para tareas de vigilancia o persecución.

La Figura 6.19 muestra a grandes rasgos el esquema de comunicaciones.

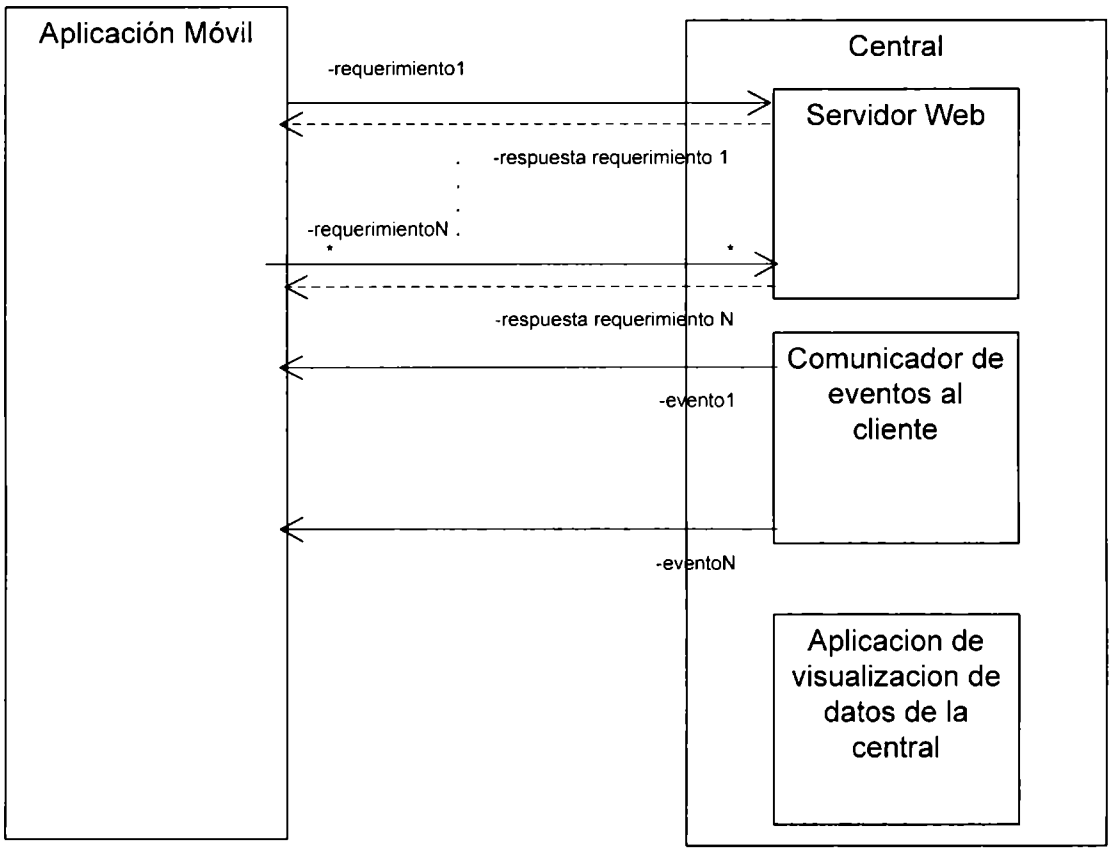


**Figura 6.19: Esquema de comunicación**

En la figura se detalla que el cliente (aplicación móvil) puede hacer requerimientos a la central y además debe estar preparado para la recepción de eventos.

### 6.4.3 Diseño de comunicación en el usuario móvil

En esta sección se presentan las clases necesarias para la comunicación de la aplicación móvil con la aplicación de la central.

Como se ha detallado anteriormente existen múltiples funcionalidades en la aplicación móvil. Es deseable que cada una de ellas sea independiente, de manera que al agregar una nueva no se modifiquen las existentes y se permita una fácil incorporación. Debido a esto cada una de las funcionalidades es implementada como un Comando (aplicación del patrón Command [Gamma, 95]).

La Figura 6.20 introduce las clases participantes para la realización de un requerimiento.

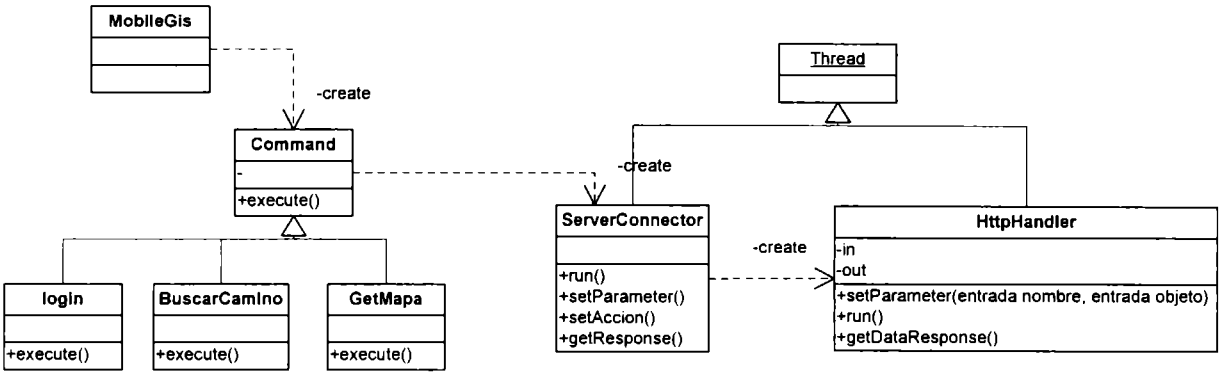


**Figura 6.20: Clases participantes en un requerimiento**

El punto de entrada a la aplicación móvil esta representada por la clase MobileGis. Cuando un usuario requiere información, en primer lugar, se crea un comando que se encargue de realizar dicha operación, por ejemplo cuando un usuario ejecuta esta aplicación lo primero que debe realizar es un login con la central, para lo cual se crea un comando de login.

Como los requerimientos deben ser procesados por la central es necesario tener alguien encargado de comunicarse con la misma. La clase ServerConnector es la responsable de entablar dicha comunicación. A un ServerConnector se le deben establecer (setear) los parámetros necesarios para la ejecución de la operación a realizar. Como se ve en la Figura 6.20 la clase ServerConnector es un Thread, por lo tanto, se crea como un hilo de ejecución aparte, debido a que la operación a realizarse puede llevar un tiempo de demora, ya sea por la operación como por el ancho de banda de la red, y se desea que el usuario no pierda el control de la aplicación.

El ServerConnector es solo un conector a la central, pero no maneja un protocolo de conexión específico. El protocolo de cómo establecer la conexión ha sido desacoplado en el objeto HttpHandler, que sabe como establecer una conexión HTTP y obtener la respuesta HTTP requerida.

La Figura 6.21 muestra una secuencia de cómo se lleva a cabo la comunicación.

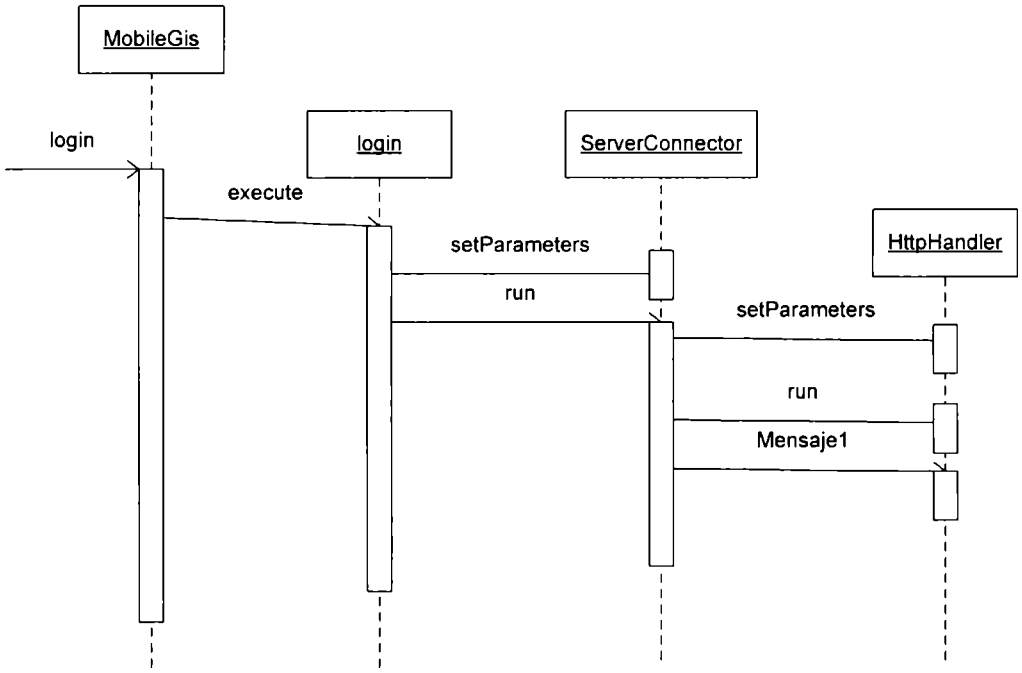


**Figura 6.21: Pedido de requerimientos desde el usuario móvil**

La figura ejemplifica a través del login la secuencia llevada a cabo cuando el usuario hace un requerimiento. Cuando un usuario intenta loguearse, se ejecuta mediante un execute un comando de login, este comando setea los parámetros necesarios para el login, entre ellos patente del vehículo, clave personal y los datos de contexto al ServerConnector específico; y le indica que se ejecute. Luego el ServerConnector crea el HttpHandler, setea los parámetros necesarios para el login, le envía el mensaje run y se queda esperando la respuesta.

El HttpHandler debe estar preparado para manejar posibles desconexiones, realizar reintentos por un período de tiempo, y en caso de no poder comunicarse enviar como respuesta que no es posible conectarse con la central.

Cuando recibe una respuesta se la envía al comando para que realice la acción pertinente en base a la respuesta.

La figura ejemplifica la operación de login pero la secuencia a seguir es idéntica para cualquier requerimiento excepto por el comando ejecutado.

### 6.4.4 Recepción de eventos en el usuario móvil

Para que la aplicación móvil sea capaz de recibir comunicaciones cuando algo ocurre existe la clase AlertsListener. Una instancia de esta clase es creada por el MobileGis como un hilo de ejecución separado una vez que se realiza la operación de login y es validado el usuario. La instancia permanece en ejecución por el tiempo de vida de la aplicación y espera por la recepción de eventos, asumiendo que recibirá para esto algún comando adecuado para realizar las operaciones apropiadas al caso. Al recibir un comando, instancia de una clase perteneciente a la jerarquía EventCommand, simplemente debe indicarsele que se ejecute. Estos EventCommand nuevamente son una aplicación del patrón de diseño Command[Gamma, 95].

La Figura 6.22 muestra una secuencia de lo pasos seguidos cuando se recibe la comunicación de que ha ocurrido un evento.

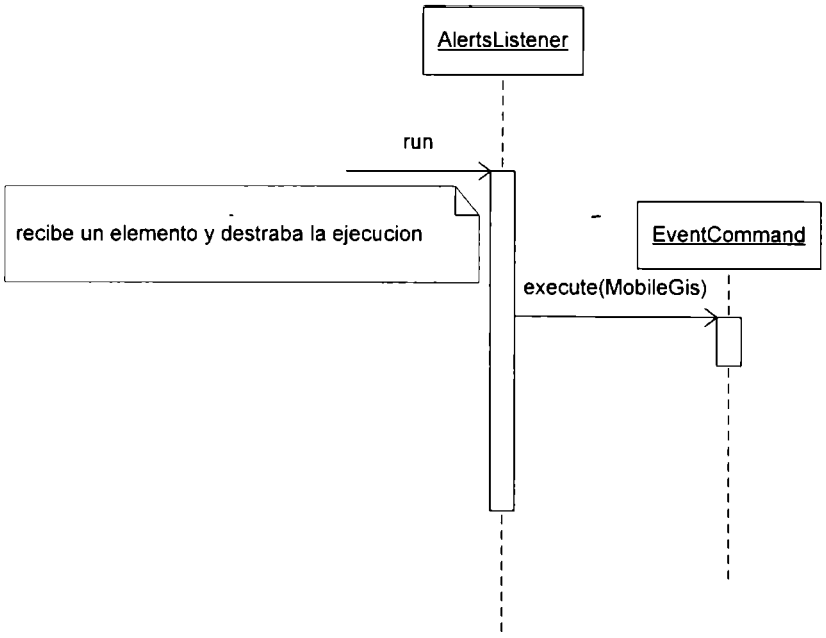


**Figura 6.22: Recepción de eventos del cliente**

En la figura 6.22 puede observarse que el objeto AlertListener se encuentra en ejecución y cuando recibe un EventCommand continúa su ejecución. Al EventCommand recibido le envía el mensaje execute, con el MobileGis como parámetro. La razón por la cual se le envía el MobileGis es para que el comando interactúe con este de forma de hacer la actualización adecuada.

### 6.4.5 Comunicación de la central con los usuarios móviles

Así como en la aplicación móvil existe una capa para establecer una comunicación con la central, en el servidor existe una capa que permite establecer la comunicación con la aplicación móvil.

Por un lado existen servidores especializados para los servicios que ofrece la central, y a los que los clientes hacen los requerimientos http. Estos servicios se encuentran disponibles a través de un servidor web. En la Figura 6.19 puede verse como los usuarios hacen requerimientos hacia el servidor web.

Dentro del servidor web existen servidores de diferentes servicios. Entre estos servidores se encuentra uno que permite registrar los patrulleros que se conectan y determinar la manera de comunicarse con los mismos, de manera que cuando ocurra un evento relevante al patrullero éste pueda ser informado.

Para poder detectar estos eventos que modifican al patrullero se crean observadores del mismo, los cuales están basados en el patrón Observer [Gamma, 95], el cual permite reaccionar ante cambios en el patrullero. Estos observadores al descubrir cambios en el patrullero crean un comando adecuado para dicho evento, le establecen

los valores adecuados y lo envían a través de la capa de comunicación (usando una instancia de ClientComunicator) al usuario móvil asociado a dicho patrullero.

La Figura 6.23 introduce las clases existentes para la comunicación del lado de la central

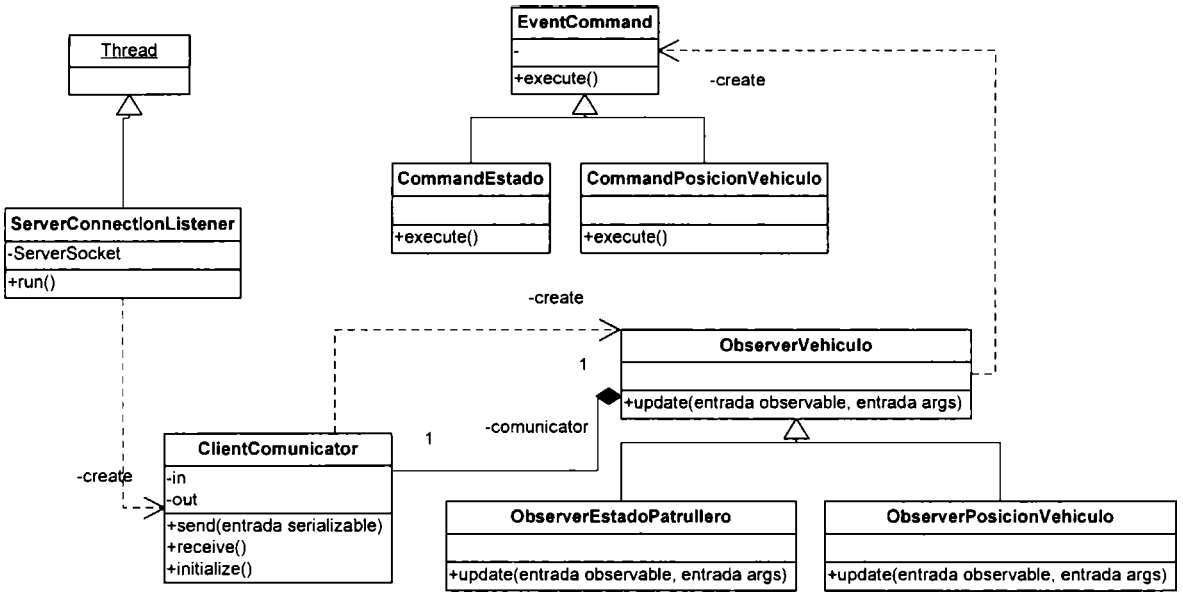


**Figura 6.23: Diagrama de clases de para comunicación en el servidor**

La clase ServerConnectionListener es la encargada de registrar cada uno de los patrulleros que se conectan a la central para enviarles información en caso de que ocurra algún evento que debe serles notificado. Una instancia de esta clase debe encontrarse corriendo para recibir dichas solicitudes mientras la central provea servicios. Dado que esta recepción no debe retardar los demás servicios es conveniente que posea un hilo de ejecución separado. Es por esto que extiende la clase Thread, la cual al ejecutarse crea un nuevo hilo de ejecución.

Cada vez que un cliente (aplicación móvil) se registra al ServerConnectionListener este crea un ClientComunicator, quien será el encargado de comunicarse con el usuario móvil específico. Además, el ClientComunicator define los observadores adecuados para el patrullero asociado al usuario móvil, de manera de poder conocer sus cambios de estado.

Cuando los observadores detectan un cambio en el patrullero crean el comando adecuado para tratar dicho evento, subclase de EventCommand (inspirado en el patrón de diseño Command [Gamma, 95]), le establecen los parámetros necesarios para realizar la tarea y lo envían a través del ClientComunicator adecuado al usuario móvil, donde se ejecutará el comando realizando las operaciones adecuadas para dicha situación.

La Figura 6.24 muestra la secuencia seguida al registrar que un usuario móvil se conecta al sistema, la cual permitirá posteriormente notificarle eventos a la aplicación de éste usuario móvil.

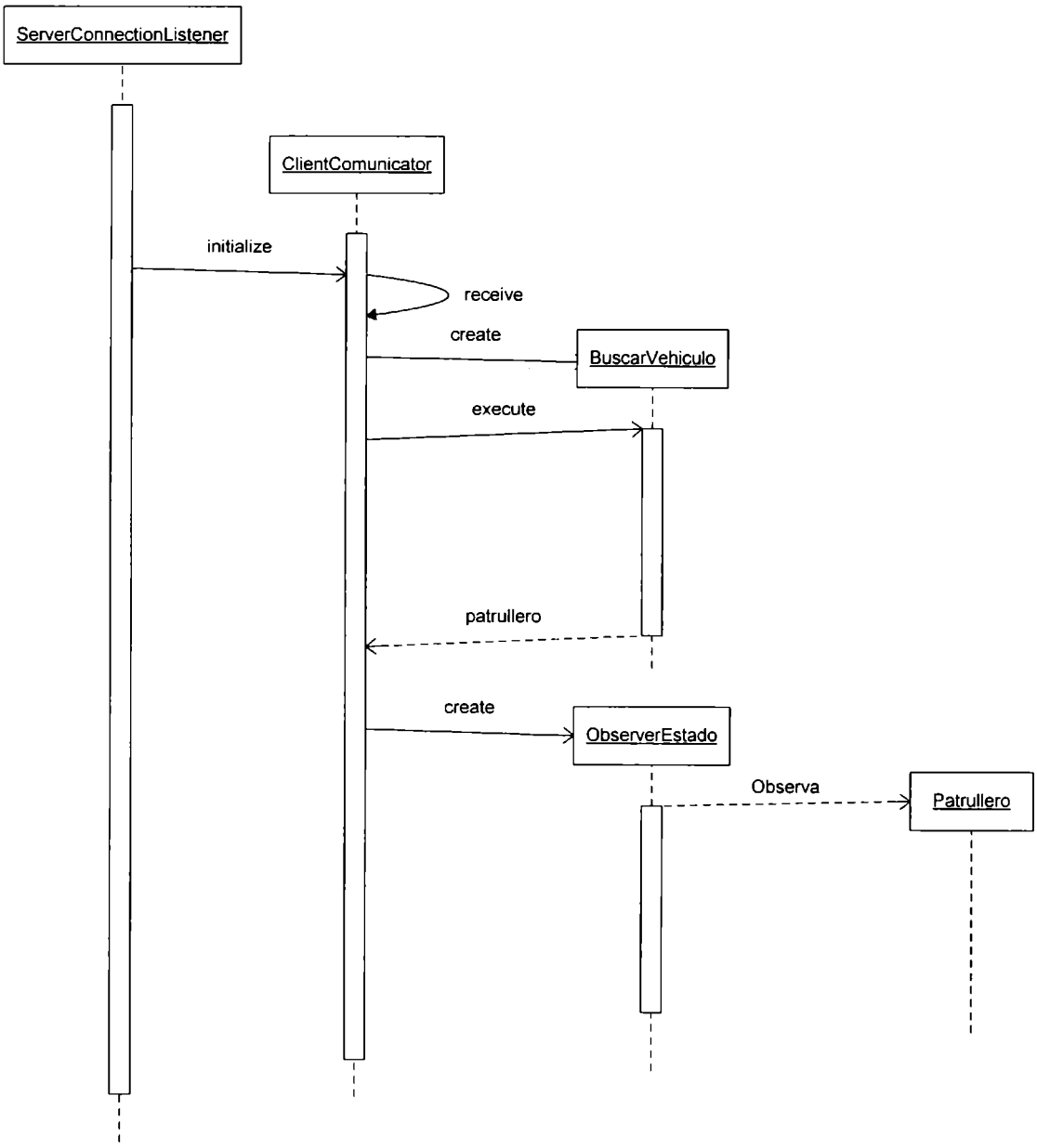


**Figura 6.24: Secuencia al conectarse un usuario al servidor**

Ampliando lo observado en la figura, se puede agregar que cuando un usuario se conecta al servidor, el ServerConnectionListener registra esta conexión y crea un ClientComunicator, donde la primer tarea de éste es recibir la patente del patrullero asociado al usuario móvil para poder conocer con cual de ellos se esta trabajando. Con dicha patente se busca el patrullero a través de un comando de búsqueda de vehículos, y crea todos los observadores necesarios para ese patrullero. En la figura se crea solo un ObserverEstado, el cual se dedicará a observar el estado del patrullero, pero pueden crearse observadores para cualquier cambio que se desee observar del patrullero.

A continuación la Figura 6.25 muestra la secuencia seguida cuando un observador reacciona ante un cambio en el estado de un patrullero, y posteriormente se desarrolla una explicación de la secuencia seguida.

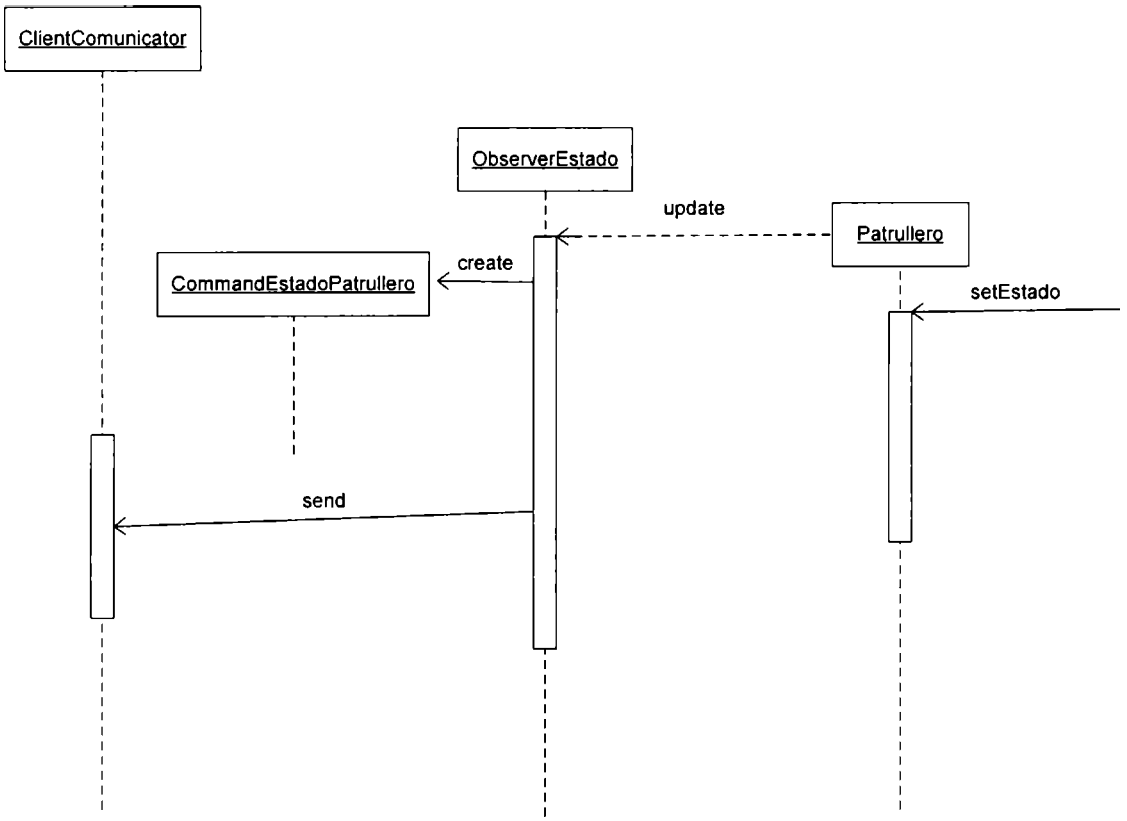


**Figura 6.25: Secuencia seguida en el cambio de estado de un patrullero**

Cuando un patrullero sufre un cambio en su estado, el observador es notificado de este cambio recibiendo el mensaje update(Object o, Object arg), donde el primer parámetro es el objeto que disparo la actualización (en este caso el patrullero) y el segundo argumentos necesarios para el caso particular. Al notificarse, éste crea un comando CommandEstadoPatrullero con la información del cambio ocurrido y le indica al ClientComunicactor que envié este EventCommand a la aplicación móvil del patrullero mediante el envío del mensaje send(Object obj), el cual tendra el efecto de vaiajar por la red hasta la aplicación móvil adecuada.

Por último, el ClienteComunicator debe estar preparado para que el envío del comando pueda no ser exitoso, hacer el reintento de envío y en caso de no poder establecer la comunicación informarlo. De ésta manera, la central es notificada de que el usuario móvil no ha podido enterarse del cambio en su estado y algún operario puede hacerlo a través de algún otro medio, por ejemplo, telefónicamente.

# 7 Implementación

## *7.1 Introducción*

El presente capítulo detalla la implementación del prototipo presentado, el cual es acotado y no contempla todas las funcionalidades planteadas en el capítulo 5. Es por ello que este capítulo describe las funcionalidades efectivamente desarrolladas.

A su vez, se describen las tecnologías utilizadas para el desarrollo y los problemas encontrados a la hora de utilizarlas en la implementación.

Además, dado que en el trabajo intervienen dos sistemas, la central y la aplicacion móvil, la implementación se detalla teniendo en cuenta a estos.

Por último, como estos sistemas se encuentran en lugares distantes y necesitan compartir objetos, se explica el mecanismo utilizado para ello.

## *7.2 Implementación de la Central*

El objetivo de esta sección es detallar la implementación de la aplicación de la central. Este prototipo es reducido y solo posee un subconjunto de las funcionalidades propuestas para una central completa.

Además de las operaciones propias de una central, esta implementación reúne operaciones de simulación, como por ejemplo el movimiento de vehículos, para poder evaluar las modificaciones que sufren los usuarios móviles con estos cambios.

### 7.2.1 Funcionalidades existentes en la implementación de la central

En esta sección se detallan las funcionalidades desarrolladas para este prototipo de la central, donde el detalle de las funcionalidades ya fue explicado en el capítulo 5

1) **Posicionar los vehículos en el mapa para visualizar la posición tanto de los vehículos monitoreados como de los patrulleros.**

2) **Asignar el robo de un vehículo a los usuarios móviles más cercanos al hecho.**

3) **Elegir que capas de información se desea visualizar**

4) **Información particular de vehículos y sus posibles conductores.**

5) **Ingresar información de vehículos y datos relacionados a ellos en el sistema.**

6) **Suministrar la información de un punto**

7) **Informar la recuperación de un vehículo a los usuarios móviles involucrados en la persecución.**

8) **Funcionalidades de manejo de visualización de mapas como zoom, pan etc.**

### 7.2.2 Funcionalidades de simulación existentes en la central

**1) Simular el movimiento de vehículos**

Dado que no se cuenta con dispositivos y autos reales que permitan la captura de eventos de desplazamientos sobre el mapa, fue necesario realizar una simulación de los mismos. La simulación del movimiento podría realizarse de dos maneras:

- Manualmente: seleccionando un vehículo y ubicándolo en una nueva posición
- Automáticamente: realizando recorridos aleatorios o preestablecidos sobre el mapa.

En este trabaja se ha optado por la alternativa 1 dado que permite realizar más casos de prueba y verificar fácilmente el correcto funcionamiento de la aplicación móvil.

### 7.2.3 Características de la implementación de la central

La central fue desarrollada sobre la plataforma J2SE (Java 2 Standard Edition), ejecutándose bajo el entorno de un servidor web (Tomcat). Esta restricción se efectúa debido a que permite soportar servicios a través de requerimientos HTTP, de manera natural para los usuarios móviles. A su vez, para el manejo y visualización de la cartografía fue utilizada la suite de componentes Map Object para Java5, dado que es un producto reconocido para aplicaciones GIS, soporta la cartografía obtenida, y provee una versión de prueba gratuita para ambientes Java.

### 7.2.4 Esquema de trabajo de la central

La central se encuentra implementada bajo el esquema de trabajo que propone "Model View Controller" [MVC].

En el esquema MVC, el flujo de la aplicación es mediado por el controller. El controller delega requerimientos a un manejador adecuado. El manejador se encuentra relacionado al modelo y cada uno de ellos actúa como un adaptador entre el requerimiento y el modelo. El modelo representa o encapsula la lógica de la aplicación, en cambio, el controller es quien decide la vista apropiada. Este esquema logra una perdida de acoplamiento entre la vista y el modelo, lo cual hace que las aplicaciones sean fáciles de crear y mantener.

#### 7.2.4.1 Modelo

El modelo del sistema esta representado básicamente por el modelo de base AVL, descripto en la sección 6.2, y los comandos que ejecutan acciones sobre él.

Para el prototipo solo fueron implementadas las clases necesarias para proveer las funcionalidades detalladas anteriormente, que reducen la funcionalidad original. Por ejemplo, este prototipo no hace control de recorridos realizados por los vehículos, por lo cual no fueron implementadas las clases intervinientes para dicha funcionalidad.

En la siguiente lista se encuentran detallados los comandos que ejecutan acciones sobre el modelo de base AVL:

---

[5] Para obtener mayor información del producto ver Apéndice II.

- Buscar vehículo
- Mover vehículo
- Robar vehículo
- Recuperación de vehículo
- Validar Login
- Agregar Empleado
- Agregar Vehículo
- Agregar Patrullero
- Agregar Conductor de vehículo.
- Buscar intersección de calles
- Buscar un robo de vehículo
- Buscar una recuperación de un robo.

Estos fueron implementados utilizando el patrón de diseño Command [Gamma, 95] dado que permiten agregar nuevas funcionalidades sin modificar el modelo existente.

#### 7.2.4.2 Modelo de controladores de la central

La capa controller esta focalizada en recibir requerimientos desde la interfaz y decidir que funcionalidad ejecutar. Los controllers fueron creados para separar, en la central, la interfaz del modelo.

En esta implementación existen dos clases diferentes de controllers:

- Los que relacionan la interfaz con el modelo.
- Los que relacionan las solicitudes y respuestas de los usuarios móviles con el modelo. El funcionamiento de estos será explicado posteriormente en la sección 7.2.8 "Servicios de la central".

#### 7.2.4.3 UI de la central

La interfaz de usuario de la central fue desarrollada como una ventana de aplicación desktop, utilizando componentes swing de java [SWING] y MapObject-java [Map Object] para la visualización de los mapas.

Algunos de los objetivos que se intentaron satisfacer con la interfaz de la central fueron:

- Permitir la visualización de todos los autos del sistema: La interfaz de la central debe contener el mapa total de la ciudad de forma que permita ver todos los vehículos que se encuentran recorriendo la ciudad. Además, en esta interfaz el área que contiene el mapa debe poseer grandes dimensiones para permitir una correcta visualización.
  La Figura 7.1 muestra la ventana principal de la central, donde pueden visualizarse la totalidad de los vehículos.


**Figura 7.1: Ventana de la central**

En la figura puede observarse el mapa, en el que existen puntos que marcan vehículos que se encuentran en él, donde el color depende del tipo de vehículo (vehículo monitoreado, vehículo robado, patrullero en vigilancia, patrullero en persecución).

- Permitir visualizar diferentes capas de información: En la central pueden seleccionarse los layer de información que desean visualizarse, lo cual permite focalizar la atención en la información adecuada.
  Además, dependiendo la escala será la cantidad de datos visualizados de los layers, por ejemplo, a una escala de visualización elevada pueden observarse los nombres de las calles de la ciudad. Esta decisión se debe a que en escalas pequeñas demasiada información resulta confusa y puede superponerse.
  La figura 7.2 muestra que a diferentes escalas de visualización pueden observarse mayor o menor información.

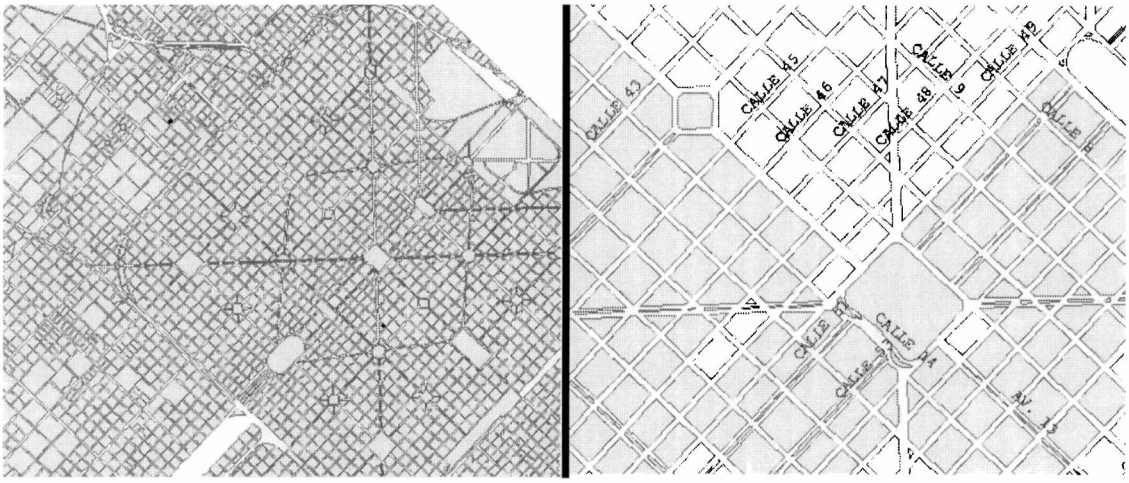


**Figura 7.2: Mapa a diferentes escalas de visualización**

En la figura puede observarse sobre el mapa que se encuentra a la derecha las calles con sus respectivos nombres, en cambio en el de la izquierda no se visualizan estos detalles.

Las diferentes capas de información que pueden visualizarse se encuentran a la izquierda de la ventana, como es observado en la Figura 7.1, encontrándose tildadas aquellas que actualmente se encuentran visibles, pudiendo seleccionarse o deseleccionarse las que se deseen.

- Permitir manipular la visualización del mapa: la interfaz de la central debe proveer características de manipulación de mapas, las cuales se encuentran disponibles en la barra de herramientas (Figura 7.3) o en la opción del menú "Ver". Ejemplos de operaciones de manipulación son zoom in, zoom out, pan, vista anterior, vista siguiente, vista completa etc.

**Figura 7.3: Opciones de visualización de la barra de herramientas**

Puede observarse en la Figura 7.3 las diferentes opciones de manipulación del mapa de la barra de herramientas.

Las opciones que se muestran en la figura son (de izquierda a derecha): vista anterior, vista siguiente, zoom al layer activo, vista completa, zoom in, zoom out, pan, pan en dirección e información sobre puntos del mapa.

Estas opciones también pueden ser accedidas mediante la opción de menú "Ver", como ya fue mencionado.

- Permitir ver datos de los autos existentes en el sistema y sus dueños o conductores.

  Para visualizar datos de un vehículo se tienen dos opciones:

  - Seleccionar el botón de la barra de herramientas ( ) o desde el menú "Ver" la opción "Seleccionar información de vehículo", y luego clickear sobre algún vehículo que se encuentra en el mapa.
  - Seleccionar con un doble clic un vehículo de los listados que pueden observarse a la derecha de la Figura 7.1.

Al seleccionar vehículo se despliega una ventana con la información asociada. En la Figura 7.4 se observa dicha ventana.

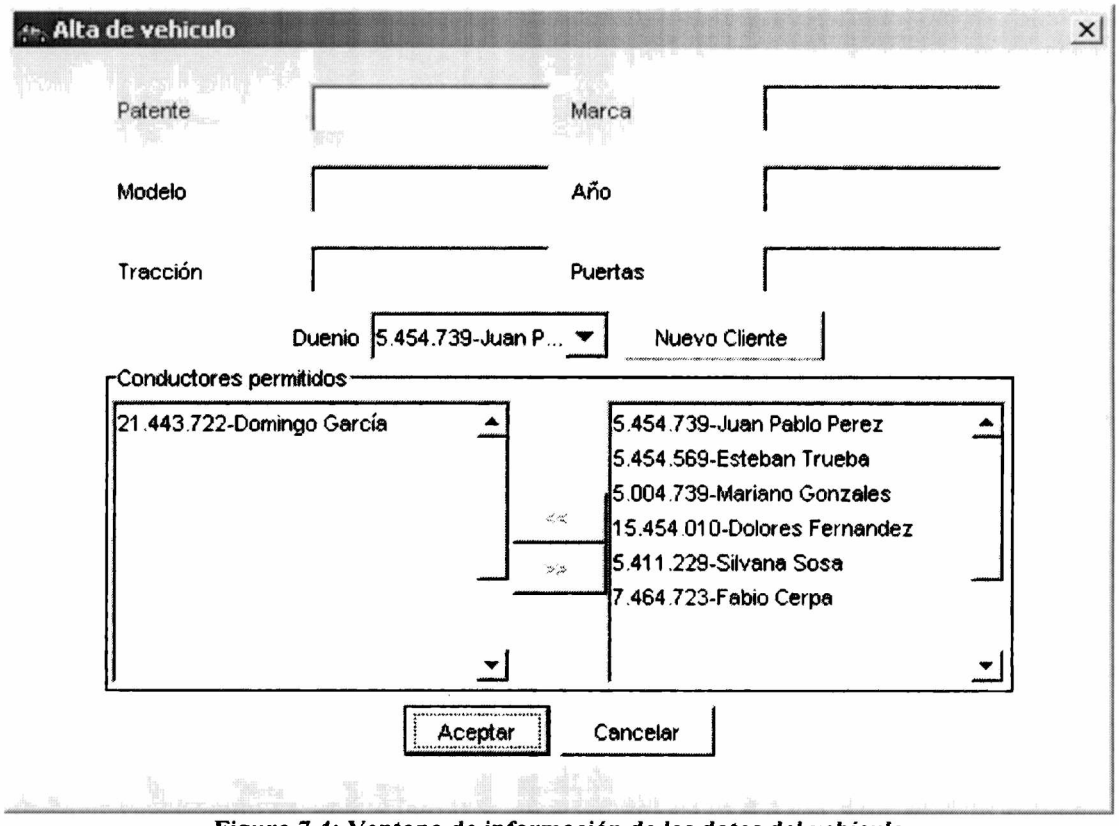


**Figura 7.4: Ventana de información de los datos del vehículo**

En la figura pueden verse los datos del auto, el dueño del mismo y las personas que posiblemente lo conduzcan.

- Permitir registrar denuncias de robos de vehículos: para informar el robo de un vehículo existen dos opciones, seleccionar el botón de la barra de herramientas ( ), o desde la opción de menú "Informar evento" el ítem "Robo de Vehículo".

  La Figura 7.5 muestra la ventana desplegada para informar el evento de robo, en la cual debe ingresarse la patente del vehículo sustraído y la fecha del robo.

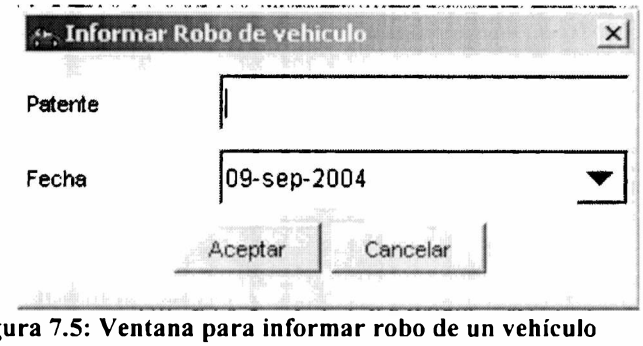


**Figura 7.5: Ventana para informar robo de un vehículo**

Al informar que un vehículo ha sido notificado como robado este cambia su color de visualización a rojo y, los patrulleros asignados a la recuperación a

verde. Dada la política de asignación de patrulleros a un robo implementada para este prototipo, los patrulleros asignados serán los dos más cercanos al vehículo sustraído. La Figura 7.7 muestra el cambio producido al informar de un robo.

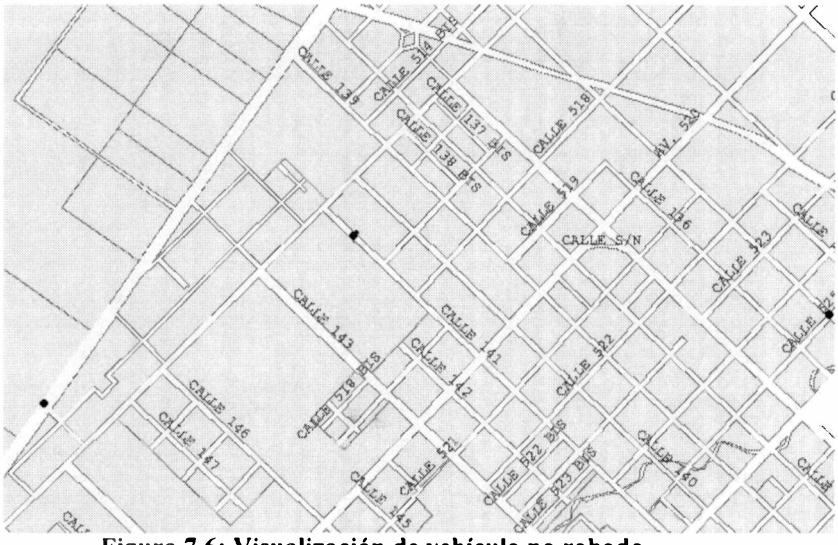


**Figura 7.6: Visualización de vehículo no robado**


**Figura 7.7: Visualización vehículo robado**

Las figuras muestran la diferente visualización que se produce en los vehículos cuando se informa el evento de robo.

- Permitir registrar recuperaciones de vehículos: para informar la recuperación de un vehículo existen dos opciones, seleccionar el botón de la barra de herramientas ( ), o desde la opción de menú "Informar evento" el ítem "Recuperación de Vehículo".
  La Figura 7.8 muestra la ventana desplegada para informar la recuperación de un vehículo robado.

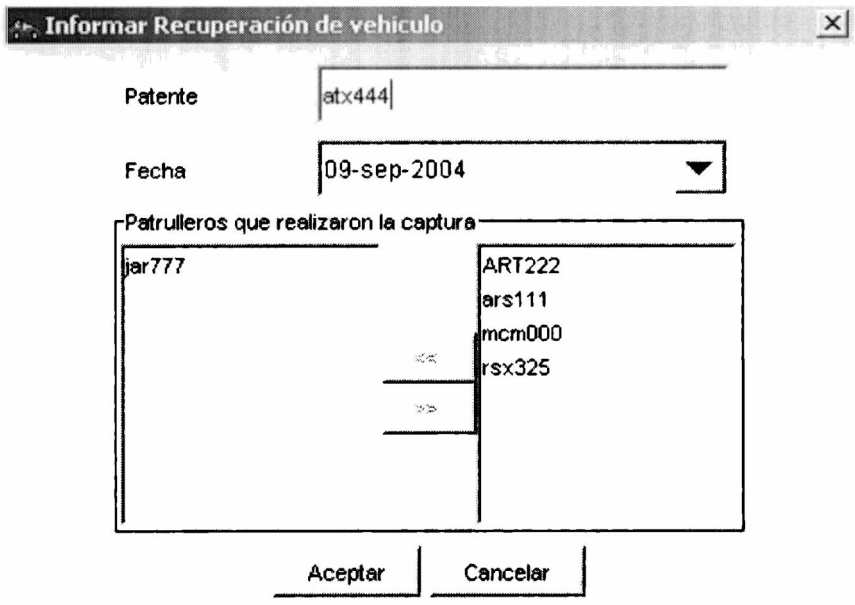


**Figura 7.8: Ventana para informar la recuperación de un vehículo**

En esta ventana debe informarse que vehículo fue recuperado, en que fecha, y los patrulleros que efectuaron su recuperación.

Cabe destacar que son ingresados los patrulleros que efectuaron la captura ya que pueden no ser todos los que tenían asignado su recuperación los que efectivamente la realizaron.

Cuando una recuperación es efectuada puede observarse sobre el mapa un cambio en la visualización de los vehículos participantes, donde el vehículo monitoreado vuelve a verse en color azul y los vehículos asignados a la persecución a azul oscuro.

- Permitir hacer operaciones de simulaciones de movimiento de vehículos: dentro del prototipo de la central se encuentran disponibles las funcionalidades de simulación de movimientos de los vehículos.

  Para realizar el movimiento de un vehículo se debe:
  - Seleccionar de la barra de herramientas las opciones mover vehículo del mapa , mover patrullero del mapa, mover vehículo robado del mapa, mover patrullero en persecución del mapa (se corresponde con las siguientes opciones respectivamente  )
  - Seleccionar un vehículo del respectivo tipo.

  La Figura 7.9 muestra como seleccionar un vehículo.

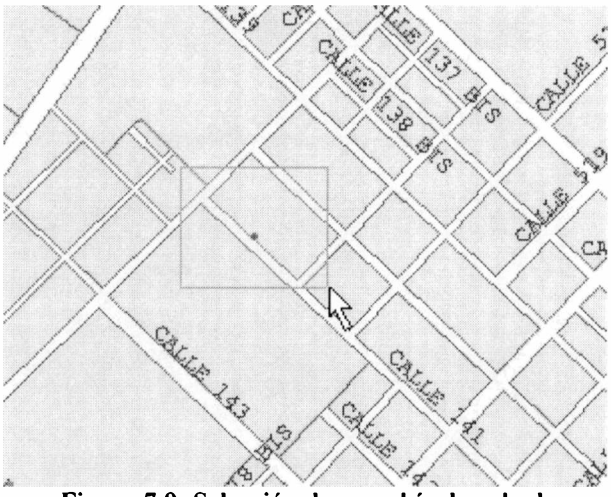


**Figura 7.9: Selección de un vehículo robado**

En la figura puede observarse como se selecciona un vehículo robado. La selección se ejemplifica con este tipo de vehículo, pero es idéntica para todos los tipos (vehículo no robado, patrullero en vigilancia, vehículo robado, patrullero en persecución).

Una vez seleccionado el vehículo que se desea desplazar este se mostrará resaltado en color amarillo.

La Figura 7.10 muestra un vehículo resaltado por la selección.

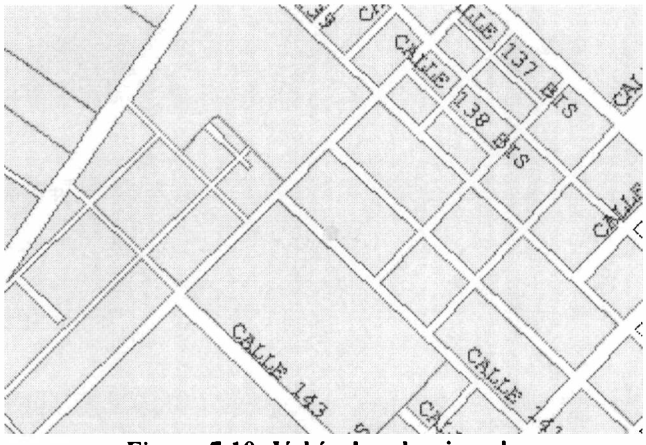


**Figura 7.10: Vehículo seleccionado**

- Mover el vehículo a una nueva posición

La Figura 7.11 muestra como mover el vehículo seleccionado en el paso previo

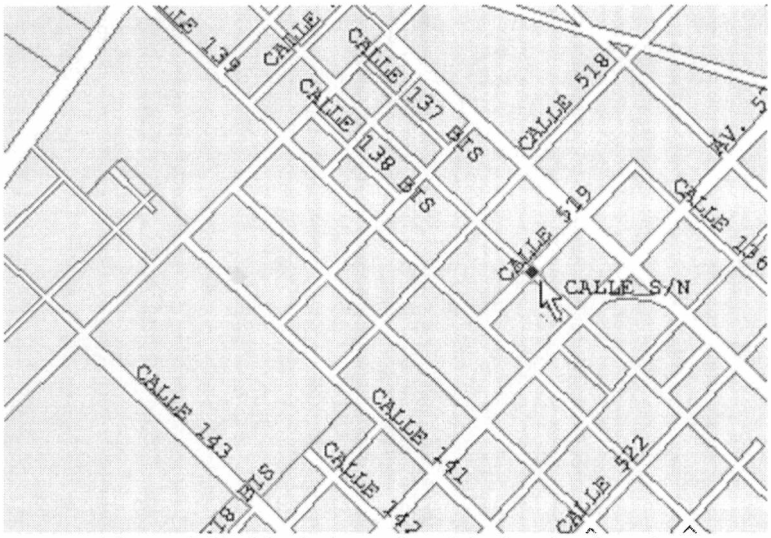


**Figura 7.11: Movimiento de vehículo seleccionado**

En la figura puede observarse un punto rojo, seleccionado con el cursor, que es el reflejo de arrastrar el vehículo seleccionado hacia una nueva ubicación. Cuando el punto es soltado en un lugar del mapa válido (una calle) el vehículo seleccionado pasa a moverse al nuevo punto, este cambio es reflejado en la Figura 7.12.

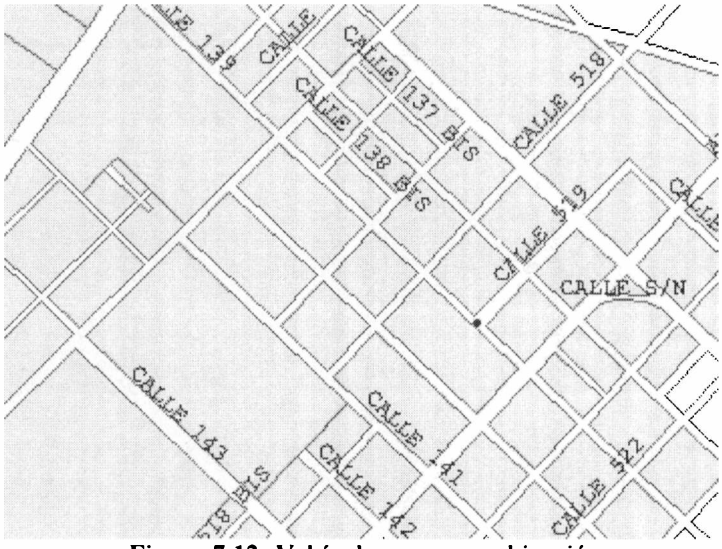


**Figura 7.12: Vehículo en nueva ubicación**

En la figura se puede contemplar la nueva ubicación del vehículo.

### 7.2.5 Servicios de la central

Los servicios que provee la central a los usuarios móviles son implementados como Actions del framework Struts, el cual esta basado en Servlets. Para entender y conocer el funcionamiento del framework Struts ver [Struts]. Cada vez que un usuario hace un requerimiento será atendido por un Action específico, quien es el encargado de atender dicho requerimiento y retornar una respuesta adecuada.

Los Action creados interactúan con el AVL para obtener la información.

La Figura 7.13 ejemplifica como interactúa un Action con el modelo AVL.

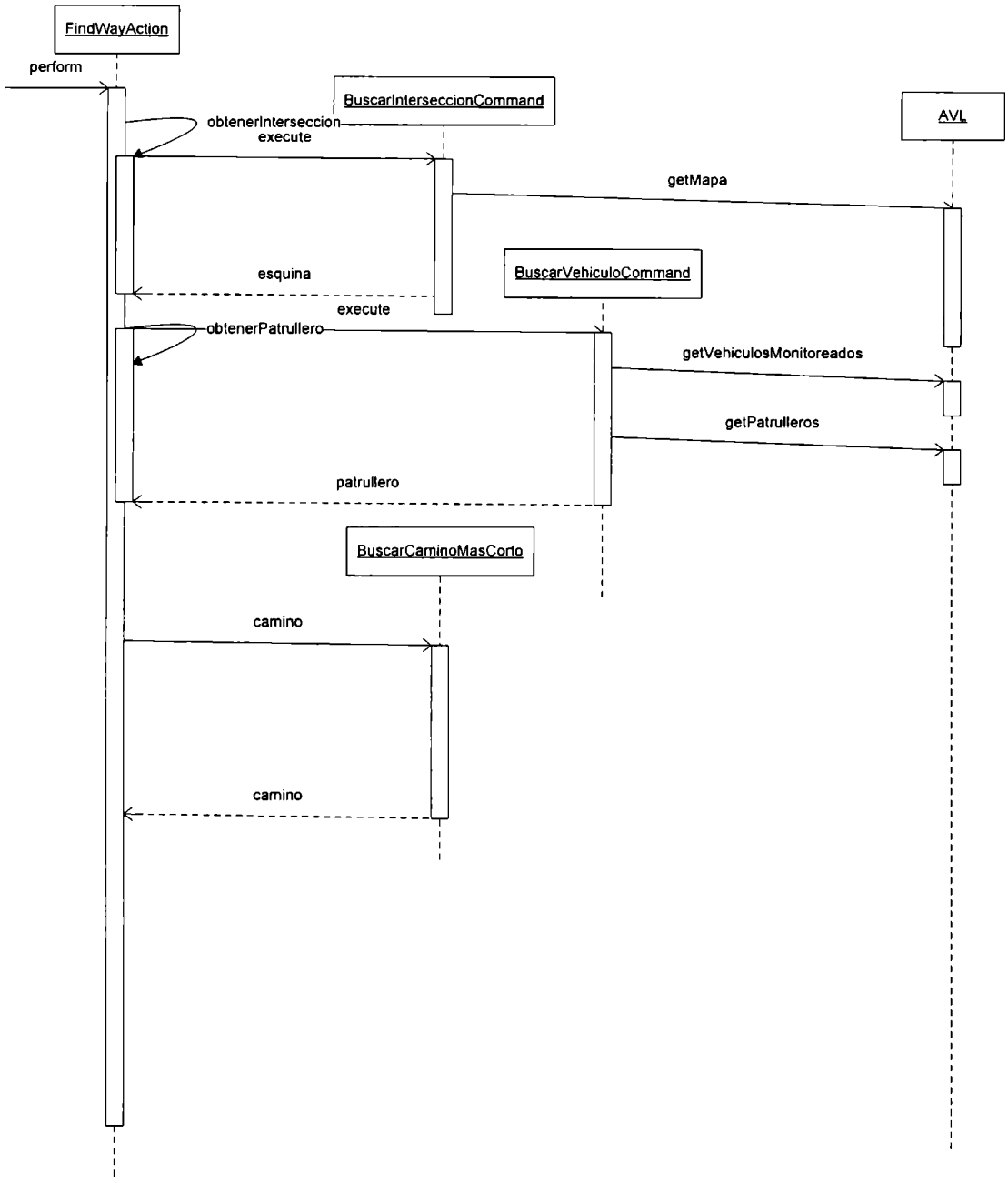


**Figura 7.13: Interacción entre una Action y el modelo de AVL.**

En la figura puede observarse el método perform de findWayAction. Este Action es el correspondiente a la atención de una búsqueda de caminos, el cual recibe dos calles (las cuales supuestamente corresponden a una intersección) y la patente del vehículo que esta haciendo el requerimiento. Primero intenta obtener la intersección, esto es realizado enviándose el mensaje obtenerInterseccion a sí mismo. En la ejecución del mismo se crea un Comando para la búsqueda de Intersección (BuscarInterseccionCommand), el cual pide el mapa al AVL y busca la intersección. Si esta intersección de calles no es encontrada en el mapa con el que se esta trabajando, retornara como resultado que la intersección no existe. Caso contrario, luego de realizado este paso se debe obtener el patrullero que hace el requerimiento, para de esta

forma obtener su posición. La realización de este paso se hace mediante el llamado al método obtenerPatrullero que crea un comando para obtener el vehículo (BuscarVehiculoCommand). Este comando solicita los vehículos al AVL, y retorna el que coincide con la patente. A partir de este punto pasará a buscarse el camino.

Cabe destacar que la figura solo ejemplifica las interacciones con el AVL, y no es el diagrama de secuencia completo a partir del método perform, dado que luego de interactuar con el AVL, este debe armar la respuesta al requerimiento, y mostrar toda esta secuencia resulta engorroso para este ejemplo.

### 7.2.6 Búsqueda de caminos

Una funcionalidad interesante que se busco proveerle a los usuarios móviles es la búsqueda de un camino a un punto. Para hacer esta búsqueda se debía trabajar con la cartografía, la cual posee gran cantidad de tramos y de cada uno de ellos gran cantidad de información, con la desventaja de que los tramos no se encuentran estructurados en forma de grafo. Los tramos tienen una posición dentro del mapa, pero no poseen información relacionada a cual es el tramo por el que pueden continuar, la única forma de obtener la relación entre un tramo y sus siguientes es a través de una búsqueda espacial sobre la cartografía, lo cual conlleva una demora importante.

Todo esto hace que realizar una búsqueda de un camino sobre el mapa (la cual ya de por si es costosa) lleve, dependiendo del caso, mucho tiempo e incluso horas. Con estos tiempos de respuesta esta funcionalidad era totalmente inútil, por lo que debieron evaluarse alternativas para solucionar el problema de performance.

Para solucionar este problema se ataco desde dos puntos:

- Solucionar la vecindad de las calles de la cartografía.
- Optimizar el algoritmo de búsqueda para encontrar el camino.

#### 7.2.6.1 Vecindad de las calles en la cartografía

Para solucionar este problema se realizo un algoritmo que generara a partir de la cartografía el grafo que representa el mapa de la ciudad. La idea fue que cuando la aplicación comenzara este grafo se llevara a memoria de forma que posteriormente las búsquedas se realizarán en ésta, lo cual resulta más eficiente. Igualmente este proceso de creación del grafo llevaba horas, lo que seguía siendo inaceptable.

Por éste motivo se debió evaluar como levantar rápidamente el grafo a memoria. Una alternativa hallada fue hacer un proceso que a partir de la cartografía generará el grafo por única vez, y luego lo guardara en algún archivo para su posterior recuperación.

Un mecanismo simple para la escritura y recuperación de objetos es la serialización, dado que solo debe indicarsele a un objeto que se guarde o recupere y el mismo sabe como realizarlo. Una vez guardado el grafo se procedió a verificar el tiempo que tardaba en recuperarse el mismo cuando la aplicación se iniciara. Esto demora tan solo unos segundos y mantener el grafo en memoria acelera notablemente la búsqueda, por lo cual con este mecanismo se alcanzo una performance aceptable para solucionar este primer punto del problema.

#### 7.2.6.2 Algoritmo de búsqueda para encontrar el camino

A partir del grafo de la ciudad obtenido en la sección anterior se comenzó a realizar un algoritmo de búsqueda de caminos sobre este. Para el prototipo solo fue implementada la búsqueda del camino más corto, considerando más corto aquel que posea la menor cantidad de tramos.

Como primer algoritmo para realizar dicha búsqueda surgió de manera natural realizar una búsqueda en profundidad sobre el grafo, la cual hace uso de recursión para determinar el camino.

No obstante, como consecuencia de la recursión la JVM (Java Virtual Machine) arrojaba un error de alocación de memoria, puesto que si el grafo posee una gran cantidad de tramos (en el caso de la ciudad de La Plata 22878), y por cada tramo que no se sabe el destino se debe entrar en recursión (debiendo alocarse memoria para este nuevo proceso), la cantidad de memoria que la JVM puede alocar era sobrepasada.

Nuevamente fue necesario evaluar una manera de solucionar este problema surgido, y una posible alternativa fue a través de una búsqueda en amplitud, la cual no utiliza recursión, y permite solucionar este problema.

Como conclusión las restricciones encontradas en la JVM deben considerarse a la hora de crear una nueva estrategia de búsqueda.

### 7.2.7 Comunicación

Dado que se requiere que la comunicación se establezca en ambas direcciones, desde la central a la aplicación móvil y desde la aplicación móvil a la central, a continuación se detalla la implementación realizada en la central para poder realizar estas comunicaciones.

- Comunicación desde la aplicación móvil a la central

  Para realizar la implementación de la comunicación entre los usuarios móviles y la central se debió, en primer lugar, decidir como implementar los servicios que brinda la central a los usuarios móviles, esta decisión ya fue explicada en la sección 7.2.8 "Servicios de la central", la cual utiliza Actions del framework Struts quienes atienden requerimientos http.

- Comunicación desde la central a la aplicación móvil

  Para que la central pudiese comunicarse con las aplicaciones móviles fue necesario implementar las clases correspondientes al diseño de comunicaciones de la central explicado en la sección 6.4.5 "Comunicación de la central con los usuarios móviles", donde el mecanismo de comunicación utilizado fue socket. Este fue escogido dado que es un mecanismo de comunicación simple soportado por la mayoría de los dispositivos móviles.

Por último dado que en el diseño planteado se requiere la transmisión de objetos entre las aplicaciones (central y móvil), se necesitaba algún mecanismo que lo permitiera. Nuevamente, debido a las restricciones de la plataforma en la que corre la aplicación móvil, el mecanismo que permitiera transmitir objetos debía ser sencillo de forma que fuera soportado por todos los dispositivos. Por este motivo se decidió hacer

uso de un mecanismo ampliamente conocido en J2SE, la serialización de objetos, pero dado que ni siquiera este mecanismo era soportado por J2ME se debió reimplementar el mismo como lo explica la sección 7.4.

## *7.3 Aplicación móvil*

El objetivo de esta sección es detallar la implementación del prototipo de la aplicación móvil, desarrollando cuales fueron las funcionalidades implementadas junto con los problemas hallados a la hora de realizar dicha funcionalidad, y la manera encontrada para poder solucionarlos.

Para el desarrollo de la aplicación móvil fue utilizada la plataforma de programación J2ME (Java 2 Micro Edition), la cual esta orientada a pequeños dispositivos tales como teléfonos móviles y PDAs, con capacidades de memoria y procesamiento limitadas y pantallas reducidas.

### 7.3.1 Funcionalidades existentes en la implementación de la aplicación móvil

En esta sección se detallan las funcionalidades implementadas para el prototipo de la aplicación móvil, siendo el detalle de cada funcionalidad ya detallado en el capítulo 5.

1. Manejo de visualización de mapas.
2. Solicitar capas de información a visualizar.
3. Solicitar el camino más corto a un punto.
4. Enviar información de contexto para que la central determine que datos enviarle.

En las secciones siguientes se profundizará en la implementación de cada funcionalidad y se mostrará la manera de interactuar con la aplicación móvil desarrollada.

#### 7.3.1.1 Manejo de visualización de mapas

Como ya se ha mencionado, los dispositivos móviles carecen de memoria y capacidad de procesamiento, así como también las pantallas que ellos poseen son de tamaño extremadamente pequeño, en particular en teléfonos celulares. Por este motivo, dichos dispositivos móviles no pueden trabajar directamente con la cartografía y realizar las operaciones sobre mapas utilizando la suite de componentes Map Object (Apéndice II), la cual requiere niveles importantes de memoria y procesamiento.

Ante estas restricciones hubo que buscar algún mecanismo que permitiera igualmente visualizar e interactuar con mapas en estos dispositivos. Un mecanismo encontrado fue hacer que la central se encargara de poseer el mapa y ante una solicitud de un usuario móvil esta realizará la operación requerida (pan, zoom, etc.), generará una imagen con el resultado del procesamiento y la enviase a la aplicación móvil como respuesta a dicha solicitud. De esta manera, se obtuvo el resultado esperado que era el manejo y visualización de mapas digitales.

Esta implementación requirió que la aplicación móvil solo implementara la metodología necesaria para pedir y recibir la vista requerida, pero que la operatoria en su totalidad sobre el mapa fuese realizada en la central.

La Figura 7.14 muestra la visualización del mapa en diferentes dispositivos.


**Figura 7.14: Visualización del mapa en los dispositivos móviles**

Las operaciones que se pueden realizar se encuentran en el menú de la aplicación, las cuales están disponibles cuando se está visualizando el mapa. En la Figura 7.15 se muestra el menú que es desplegado.

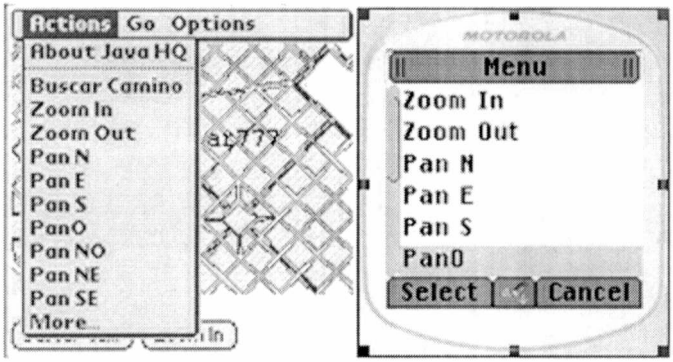


**Figura 7.15: Menú con las operaciones sobre el mapa.**

En el caso de teléfonos celulares, estas operaciones pueden ser ejecutadas presionando alguno de los botones. A continuación se detalla que funcionalidad esta asociada a cada botón:

1: Pan Noroeste
2: Pan Norte
3: Pan Noreste
4: Pan Oeste
5: Zoom In
6: Pan Este
7: Pan Suroeste
8: Pan Sur
9: Pan Sureste
*: Zoom a la extension completa (Full Extents).
0: Zoom Out
#: Zoom a la extensión original (Initial Extents).

### 7.3.1.2 Solicitud de capas de información a visualizar

Otra funcionalidad importante es que el usuario pueda seleccionar que capas de información desea visualizar. En el caso de este prototipo solo pueden seleccionarse dos capas de información: calles y manzanas. La selección de las capas deseadas se hace simplemente seleccionando o no cada capa de información.

En la Figura 7.16 puede apreciarse la pantalla que permite realizar la selección de capas.

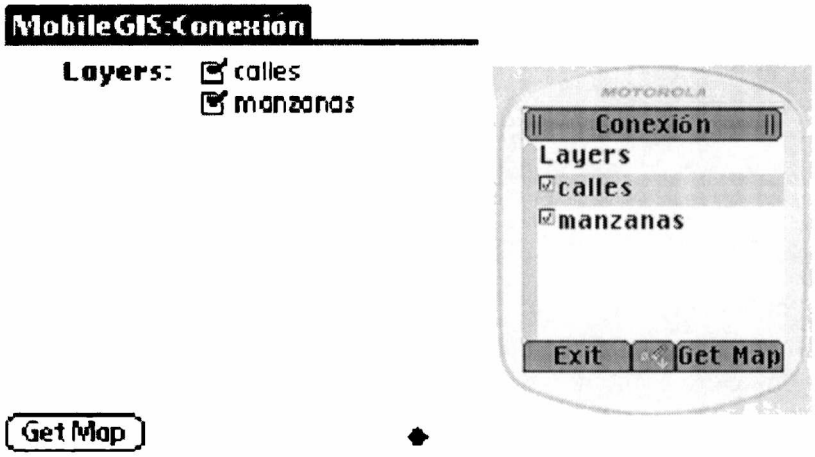


**Figura 7.16: Selección de las capas de información.**

### 7.3.1.3 Solicitar el camino mas corto a un punto.

Dado que la aplicación móvil no contiene la información necesaria para determinar el camino entre dos puntos, dada sus capacidades de almacenamiento y procesamientos limitadas, el usuario móvil deberá indicar el destino al que desea llegar, estableciendo una esquina como una intersección de calles, y requerir el camino a la central. El resultado puede ser en uno de dos modos posibles: texto (indicando el camino como una secuencia en modo texto) o gráfico (resaltando en el mapa el camino encontrado).

Los problemas encontrados para la implementación de dicha búsqueda ya fueron mencionados en la sección 7.2.9 "Búsqueda de caminos".

En la Figura 7.17 se muestra la pantalla que permite realizar la búsqueda.

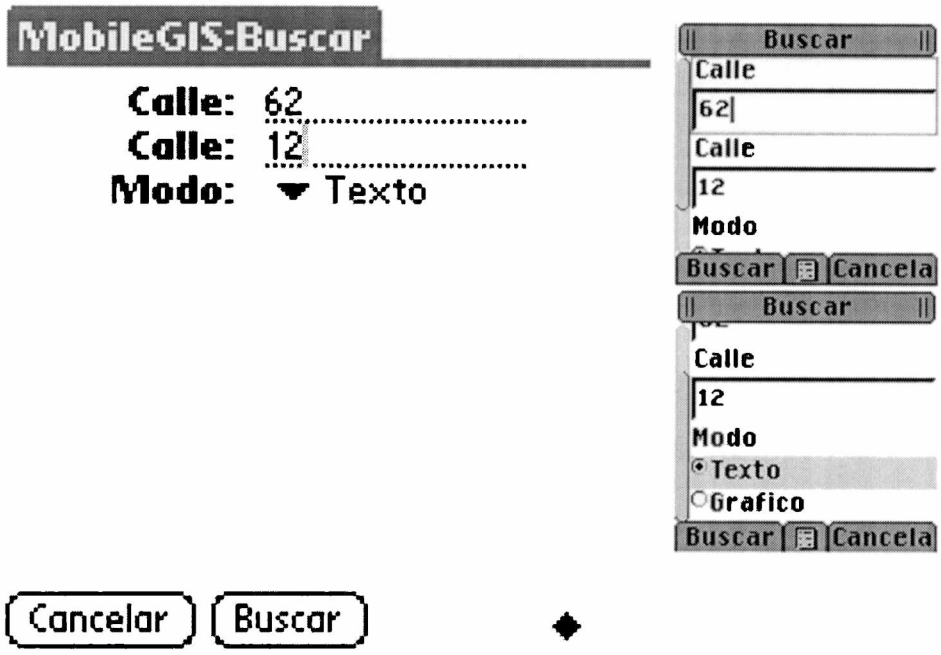


**Figura 7.17: Pantalla de búsqueda**

Como puede verse en la figura, dado que la pantalla de un teléfono celular es mucho más pequeña se debe desplazar la pantalla para poder completar todos los datos. Esto no es necesario en la PDA donde su pantalla posee mayores dimensiones.

En la Figura 7.18 se muestra el resultado de la búsqueda en modo texto.

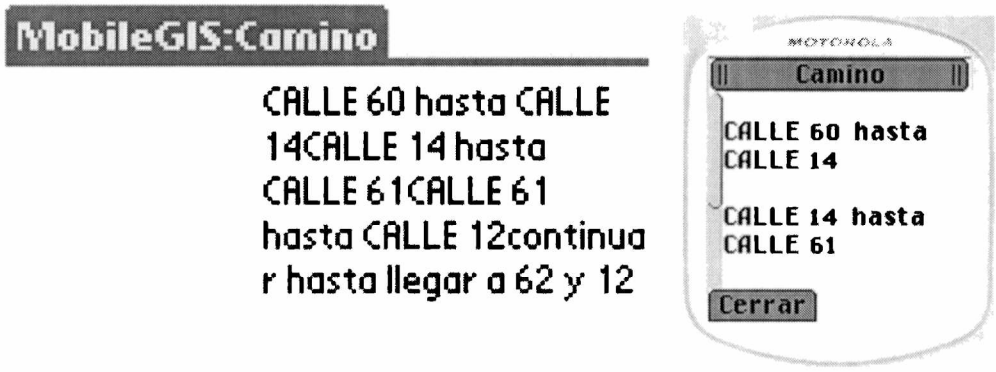


**Figura 7.18: Camino en modo texto.**

Nuevamente se puede observar que aunque la información sea poca la pantalla del celular igualmente requiere un desplazamiento. A su vez, se puede ver que la presentación de la pantalla del teléfono celular es más clara, dado que verdaderamente lista los datos, y no como la PDA que muestra cada ítem uno seguido de otro.

Por último, la Figura 7.19 muestra el resultado de la búsqueda del mismo camino en modo gráfico, utilizando un zoom adecuado para una mejor visualización.

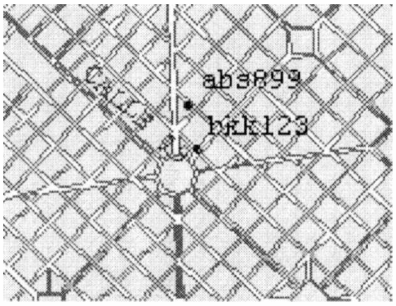

**Figura 7.19: Camino gráfico.**

En la figura puede observarse el camino encontrado (resaltado en color celeste) desde la ubicación del patrullero con patente bkk123 (usuario móvil que realizo la solicitud) hasta la intersección de las calles 62 y 12.

#### 7.3.1.4 Envió de Información de contexto a la central

Cada vez que un usuario hace requerimientos le envía a la central su contexto, en particular en este prototipo las propiedades de contexto que han sido tenidas en cuenta son: ubicación, dispositivo y estado del patrullero (vigilancia o persecución).

En cuanto a las características del dispositivo solo son contempladas las que definen la calidad de la pantalla, es decir, si la pantalla es color, su alto y ancho

### 7.3.2 Adaptaciones implementadas

Para cada una de las propiedades de contexto se realizan adaptaciones. A continuación se detalla las adaptaciones implementadas para cada propiedad de contexto:

**Dispositivo**: las características del dispositivo (color, alto y ancho) son utilizadas para adaptar la imagen que son generadas por la central como respuesta a una solicitud de mapa.
**Ubicación:** La ubicación es utilizada para determinar la zona de cobertura del mapa de la cual se debe encargar el patrullero.
**Estado:** El estado es utilizado para adaptar el tiempo de actualización del mapa visualizado en el dispositivo móvil.

Otra adaptación realizada en base al estado es la zona correspondiente al patrullero. Para el prototipo fueron implementadas dos estrategias de asignación de zonas diferentes, una para vigilancia y otra para persecución. Para el estado de vigilancia fue desarrollada una estrategia de asignación de zonas fijas, con centro en el usuario móvil con un radio preestablecido. Para el estado de persecución la estrategia de asignación de zonas se determina incluyendo el vehículo robado y todos los patrulleros asignados a su persecución.

### 7.3.3 Comunicación con la central

Para la comunicación con la central fue utilizado el diseño planteado en la sección 6.4.3 "Diseño de comunicación en el usuario móvil", en el cual se expresa que la comunicación puede ser realizada a través del protocolo HTTP, siendo la clase HttpHandler la encargada de establecer la comunicación utilizando este protocolo.

La plataforma J2ME dispone del soporte necesario para la creación de todo tipo de clientes HTTP. Estos clientes HTTP están pensados en un principio para la utilización y comunicación con aplicaciones residentes en PCs, de modo que los dispositivos móviles puedan aprovechar todos los servicios ofrecidos por estas máquinas y tengan acceso a todo tipo de información almacenada en ellas (páginas Web, descarga de aplicaciones, etc ...).

En la Figura 7.20 se muestra una secuencia común de llamadas entre un cliente y un servidor HTTP.

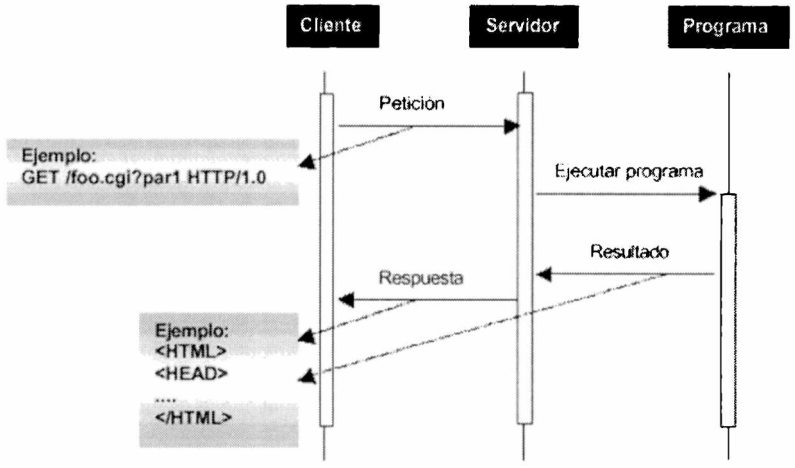


**Figura 7.20: Secuencia de llamadas entre un cliente y un servidor HTTP.**

J2ME soporta las siguientes formas de comunicación, aunque cada fabricante solamente implementa algunas de ellas:

- HTTP Connector.open("http://www.foo.com");
- Sockets Connector.open("socket://129.144.111.222:9000");
- Datagramas Connector.open("datagram://129.144.111.333");
- Puerto Serie Connector.open("comm:0;baudrate=9600");
- Archivos Connector.open("file:/foo.dat");

Existen dos especificaciones del perfil MIDP (Mobile Information Device Profile): la especificación MIDP 1.0 [MIDP 1.0] y la MIDP 2.0 [MIDP 2.0]. Cada especificación define cuales son las características que son soportadas en la misma. Por ejemplo, la MIDP 1.0 solo soporta comunicación HTTP, en cambio la MIDP 2.0 soporta todas las mencionadas anteriormente. A su vez, como ya se menciono, cada fabricante implementa solo algunas de ellas. En el caso de teléfonos celulares, hoy en día, en su gran mayoría implementan la especificación MIDP 2.0, en cambio, PALM solo implementa la especificación MIDP 1.0, por lo cual por ejemplo, sockets no es soportado.

Siguiendo con la implementación del prototipo, los requerimientos fueron implementados sin problemas a través del protocolo HTTP, dado que tanto teléfonos celulares como PDAs soportan dicho protocolo. Por otra parte, para que la central pudiese comunicarse con el dispositivo un mecanismo sencillo para lograrlo era la utilización de sockets, pero por defecto, como ya fue mencionado, no se pueden utilizar en la implementación MIDP 1.0 Aunque están implementados a nivel CLDC (Connected, Limited Device Configuration), no se tiene acceso a ellos desde el perfil, únicamente se emplean a través de determinadas clases que hacen uso de ellos, aunque nunca directamente. Debido a esto es necesario encontrar alguna solución a dicho problema.

#### 7.3.3.1 Soluciones Planteadas

Entre las diversas soluciones planteadas a la hora de utilizar sockets por diferentes programadores se pueden destacar las siguientes:

- Existe una característica no documentada que permite ejecutar un programa que utilice UDP o un socket, estableciendo para ello la variable de entorno *com.sun.midp.io.enable_extra_protocols* a true (verdadero). Esta variable se encuentra disponible en el archivo *internal.config* del directorio bin del Wireless Toolkit. Esto se debe a que los sockets se encuentran implementados a nivel CLDC, aunque no se tiene acceso directo a ellos desde el perfil MIDP.
  Activando esta variable en el Wireless ToolKit debería tener acceso a los sockets, tanto activos como pasivos. No obstante, esta solución es algo relativa, ya que en los entornos de ejecución en donde se puedan realizar este tipo de modificaciones (por ejemplo la máquina virtual para Palm OS) la utilización de sockets seguiría sin estar disponible.
  Por estos motivos esta solución solo podría ser utilizada de manera temporal, y con las pruebas realizadas no funciono correctamente, por lo cual fue descartada.

- Una segunda solución a este problema consiste en ejecutar el MIDlet con el parámetro -D indicando que se desea tener accesibles los protocolos adicionales (en caso de que existan) a nivel de perfil. Al igual que con la solución anterior, va a existir la problemática de aquéllos entornos de ejecución en donde no sea posible pasar parámetros al ejecutar los MIDlets.

- Otra posibilidad consiste en engañar a la clase *Connection* de manera que crea que el socket es un protocolo válido. Esto se puede hacer creando una clase *Protocol.class* y situándola en *com/sun/midp/io/j2me/socket*, que es donde la clase Connector buscará los protocolos soportados.
  Esta clase *Protocol* debería heredar de *com.sun.cldc.io.j2me.socket.Protocol.*
  Relacionada con esta solución está la variable *javax.microedition.io.Connector.protocolpath* del archivo *system.config* con la que se puede especificar la ruta en dónde se encuentran las clases con los protocolos disponibles en el sistema.
  Lamentablemente esta solución tampoco funcionó.

- Existe la posibilidad de recompilar el perfil con las opciones de soporte de sockets activadas de manera que el nuevo perfil soporte la utilización de estos. Este cambio habría que realizarlo para todas y cada una de las diferentes

máquinas virtuales existentes (Palm OS, etc.). Lo cual escapa del alcance de esta tesis.

- Por último, fue hallada una solución simple pero eficaz, en la cual se engaña a la aplicación haciéndola creer que nunca trabaja con un socket, dado que no se hace referencia a clases que lo mencionen, pero abriendo uno igualmente. Haciendo uso de la interfaz StreamConnection la aplicación trabaja sin percatarse de que en realidad ese Stream de datos proviene desde un socket.
  Las siguientes líneas son la única diferencia existente en trabajar con la clase SocketConnection:

```
StreamConnection sc;
sc=(StreamConnection)Connector.open("socket://127.0.0.1:5001");
```

De ésta manera utilizando la última solución planteada se pudo lograr que la central pudiese notificarle a la aplicación móvil la ocurrencia de eventos. Para esto, debe existir en la aplicación móvil un proceso en un hilo de ejecución separado que este atento al arribo de algún evento. La encargada de esto es una instancia de la clase EventsListener la cual implementa la clase Runnable y se ejecuta en un Thread aparte. Esta clase asume que recibirá el comando adecuado para el evento ocurrido y solo le indicará que se ejecute.

Como se describe, es necesario el envío de objetos a través de la red, por lo cual fue utilizado un mecanismo ampliamente conocido en J2SE, la serialización, el cual puede convertir un objeto en un flujo de bytes.

En la sección 7.4 se desarrolla el concepto de serialización en J2SE, y la forma de implementar un soporte básico del mismo en J2ME.

## *7.4 Serialización*

En este apartado se hace una breve introducción de los conceptos relativos a la serialización y la reflexión en J2SE con el objetivo de asentar las bases para un posterior desarrollo del mecanismo.

Una vez introducidos todos los conceptos necesarios se expondrá de manera detallada los pasos dados, así como los resultados obtenidos al implementar un mecanismo básico de serialización/deserialización en J2ME.

### 7.4.1 Introducción

La serialización es un mecanismo mediante el cual se puede convertir un objeto en un flujo de bytes que represente su estado, y consecuentemente ser transportado a través de la red o almacenado de manera persistente en un sistema de archivos. Esta conversión no tendría sentido si no fuese a haber una posterior recuperación, mecanismo denominado deserialización.

Tecnologías como Java (J2SE) hacen uso de esta tecnología soportando la escritura y lectura de objetos en flujos de bytes, y definiendo una serie de características que van a proteger aquella información a ser serializada.

## 7.4.2 Necesidad

Debido al intercambio de datos e información requerido entre la central y los dispositivos móviles surgió la necesidad de serialización de objetos entre ambos, la cual permite compartir estos objetos entre aplicaciones que se encuentran corriendo en diferentes lugares.

Esta necesidad obliga a crear un soporte básico de serialización en J2ME, ya que este lenguaje no incluye ninguna de las características básicas de serialización y reflexión6 provistas por J2SE. Debido a esto, se debió buscar alguna manera de conseguir serialización, de manera que se pueda almacenar el estado de los objetos de J2ME. Para conseguir este objetivo fue necesario definir una serie de interfaces y clases que de manera análoga a la arquitectura J2SE den soporte a la serialización.

Un problema que resuelve la serialización es el de poder mantener el estado de un objeto de manera persistente. Sin serialización se debe recurrir a soluciones intermedias tal como escribir de manera explícita los campos de los objetos para posteriormente volver a cargarlos.

Cuando se utiliza serialización los datos son automáticamente enviados desde una aplicación para ser recuperados por otra en el otro extremo, en donde se decidirá qué hacer con estos.

En la Figura 7.21 se muestra gráficamente el proceso de serialización y deserialización.

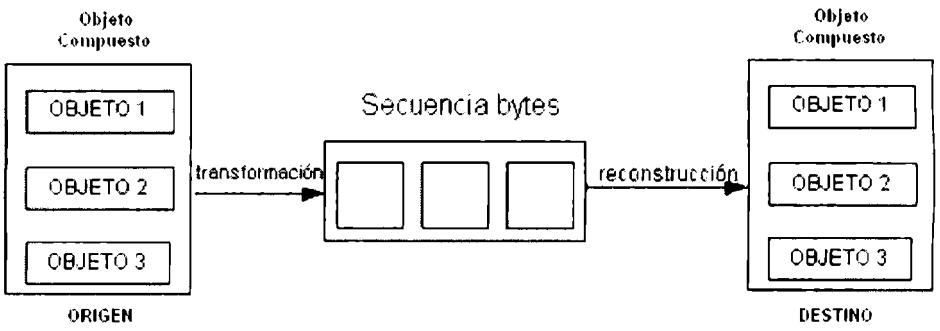


**Figura 7.21: Proceso de serialización y deserialización.**

## 7.4.3 Serialización en J2SE

A continuación se describe el proceso de serialización y reflexión en J2SE con motivo de comprender las desventajas encontradas al realizar el soporte similar en J2ME.

### 7.4.3.1 Descripción

A partir de la versión 1.1 de Java se agrego la característica conocida como serialización de objetos que permite a cualquier objeto que implementa la interfaz Serializable convertirlo en una secuencia de bytes. Con serialización se podría, por ejemplo, crear un objeto en una maquina Windows, serializarlo y enviarlo a través de la red para que una maquina Unix lo reconstruya, independientemente de la representación

---

[6] Mecanismo de introspección por el cual se puede interrogar a un objeto acerca de cuales son sus métodos y atributos.

interna de los datos o del orden de los bytes, ya que el proceso de deserealización es el que conoce como se organizan los bytes y que significado tiene cada uno.

#### 7.4.3.2 Reflexión

La reflexión es una característica del lenguaje de programación Java que permite a un programa inspeccionar clases y objetos en tiempo de ejecución.

Mediante la reflexión un programa escrito en Java puede acceder a información sobre los campos, métodos y constructores de las clases cargadas. La API de la reflexión representa las clases, objetos e interfaces que se encuentran en la Máquina Virtual Java (JVM). El paquete en el que se encuentra toda la funcionalidad de la reflexión es el java.lang.reflect.

Con la API de la reflexión se puede hacer lo siguiente:

- Determinar la clase de un objeto.
- Obtener información sobre los modificadores de las clases, campos, métodos, constructores y superclases.
- Encontrar constantes y declaraciones de métodos que pertenecen a una interfaz.
- Crear una instancia de una clase cuyo tipo no se conoce hasta el tiempo de ejecución.
- Obtener o establecer el valor de un campo de un objeto, aunque no se conozca su tipo hasta la ejecución.
- Realizar una llamada al método de una clase, desconocido en compilación.

La reflexión trabaja íntimamente relacionada con un tipo especial de objetos, los Class. Estos objetos contienen información sobre las clases (meta clases) y son empleados para crear todos los objetos regulares de nuestras clases.

Existe un objeto Class por cada clase de nuestro programa que se almacena en el archivo .class con su mismo nombre. Previamente a la ejecución, la JVM busca los objetos class necesarios y los carga en memoria.

Uno de los principales métodos de la clase Class es forName(String) que devuelve un manejador a la clase pasada como parámetro. Este manejador permite acceder mediante la reflexión a las propiedades de esa clase.

#### 7.4.3.3 Mecanismo de serialización

La clave para almacenar y posteriormente recuperar objetos en formato serializable radica en representar el estado de los objetos con la información suficiente para poder recuperarlos. Cualquier objeto susceptible de ser serializado debe implementar obligatoriamente la interfaz Serializable o Externalizable. Existen casos en los que un objeto está a su vez compuesto por otros objetos. El mecanismo de serialización debe garantizar que estos objetos son también almacenados manteniendo la relación con el que los contiene.

- **Contenedores de objetos serializados**
  Todos los objetos que actúan como contenedores deben implementar una interfaz que permite a los datos primitivos y a los objetos ser almacenados y recuperados de éste. Estas interfaces son *ObjectOutput* y *ObjectInput*.

Es tarea de estas interfaces proporcionar un flujo en dónde escribir y de dónde leer, manejar peticiones para escribir objetos en el flujo y manejar peticiones para leer objetos en el flujo.
Cada objeto que se vaya a almacenar en el flujo debe permitir explícitamente ser almacenado y recuperado, implementando para ello los protocolos necesarios (Serializable o Externalizable). Estos protocolos permiten al contenedor pedir al objeto que escriba o lea su estado.

- **Escritura de un Object Stream**
  Los pasos a seguir para escribir un objeto en un Stream son los siguientes:
  1. Crear un Stream de salida (puede ser un socket, archivo, etc.).
  2. Crear un ObjectOutputStream que es la clase encargada de serializar los objetos en Java.
  3. Realizar la serialización de los objetos mediante el método writeObject(Object) empleando para ello el ObjectOutputStream asociado a un flujo de salida.

  El método writeObject serializa el objeto especificado y recorre sus referencias a otros objetos recursivamente para crear una representación serializada completa del grafo de objetos.

- **Lectura de un Object Stream**
  Para deserializar un objeto es necesario crear un ObjectInputStream que lea del flujo de entrada en donde se encuentra el objeto.
  Generalmente los objetos se leen con el método readObject y los datos primitivos con los métodos de la clase DataInput.
  En la lectura de un Object Stream, el método readObject deserializa el siguiente objeto que encuentre en el flujo recorriendo todas las referencias recursivamente a otros objetos para crear el grafo completo de objetos serializados.

### 7.4.4 Serialización en J2ME

El lenguaje de programación J2ME no contempla características tales como la serialización o la reflexión. Su orientación a dispositivos móviles con limitaciones de memoria y capacidad de procesamiento, así como las restricciones de seguridad impuestas hacen que no se disponga de tales mecanismos.

Por este motivo se requiere conseguir un mecanismo de serialización en el que cada clase serializable sepa qué tiene que hacer para convertirse en un flujo de bytes. Para ello se deben emplear una serie de interfaces y clases que sirvan de soporte para este proceso, facilitándolo y automatizándolo en la medida de lo posible.

El programador deberá serializar manualmente sus nuevas clases creadas a partir del soporte mencionado anteriormente. Este proceso trata de simplificar al máximo de modo que solo se tengan que implementar los métodos de escritura y lectura según las interfaces definidas.

#### 7.4.4.1 Problemas existentes

El principal problema a afrontar a la hora de serializar es la imposibilidad de conocer en tiempo de ejecución los métodos y atributos de las clases. J2SE dispone del mecanismo de reflexión, que como ya se ha dicho, permite conocer las características de las clases en tiempo de ejecución.

Esta carencia hace que no pueda llevarse a cabo la serialización automática, siendo el programador el que tenga que escribir cada uno de los atributos en un flujo de bytes. La serialización llevada a cabo facilitará y se encargará de que el programador se despreocupe de cómo se lleva a cabo esta escritura, del formato interno de los datos, así como del proceso de recuperación de las clases serializadas.

Un segundo inconveniente encontrado es el hecho de que en J2SE la gran mayoría de las clases base de que dispone son serializables y disponen de los mecanismos necesarios para que se puedan serializar/deserializar. Por el contrario en J2ME no existe este concepto, no hay ninguna clase serializable, con lo que habrá que buscar algún mecanismo que permita al programador serializar sus objetos fácilmente

#### 7.4.4.2 Elementos necesarios

Si se descarta la reflexión como elemento principal a la hora de serializar, debido a que el perfil MIDP no proporciona los mecanismos necesarios para examinar las clases en tiempo de ejecución, lo siguiente más importante es disponer de algún modo de escribir y leer tipos básicos en un flujo de bytes.

Para ello J2ME proporciona las clases DataInputStream y DataOutputStream que permiten a una aplicación leer y escribir tipos primitivos Java en y de un flujo de bytes.

La clase DataOutputStream tiene métodos como writeBoolean, writeByte, writeInt, writeLong, writeShort y writeUTF que escriben datos primitivos en un flujo de bytes de manera independiente y portable. Por otro lado, la clase DataInputStream permite leer estos datos de un flujo de bytes mediante llamadas a readBoolean, readByte, readInt, readLong, readShort y readUTF.

Basándose en estas dos clases y creando las interfaces y clases necesarias se logra un mecanismo básico de serialización/deserialización que si bien necesitaría de la ayuda del programador, facilitaría en la medida de lo posible su funcionamiento.

#### 7.4.4.3 Solución planteada

En primer lugar se debió definir una interfaz Serializable para aquéllas clases que deban serializarce/deserializarce. Las clases que quieran serializarce deberán implementar dicha interfaz implementando sus dos métodos (readObject y writeObject) de modo que puedan realizar cualquiera de estas dos acciones.

A su vez, fue necesaria la creación de dos nuevas clases ObjInputStream y ObjOutputStream. Estas clases son las encargadas de controlar el proceso completo realizando las llamadas necesarias a los métodos readObject y writeObject que todas las clases serializables deben implementar.

### 7.4.5 Implementación de la serialización en J2ME

Aquí se describirá el formato de los datos internos del proceso de serialización, así como el algoritmo empleado para este fin.

#### 7.4.5.1 Formato de los datos

El primer aspecto a tener en cuenta es el formato de los datos. Al llevar a cabo el proceso de serialización los diferentes objetos son convertidos en un flujo lineal de

bytes mediante el cual posteriormente se podrán reconocer sus partes, que si bien conforman un formato interno, externamente no conservan ninguna estructura.

A grandes rasgos el formato serializado de una clase sería el siguiente:

1. Nombre de la clase.
2. Datos del primer atributo.
3. ....
4. Datos del último atributo.

Debido a que cualquiera de los atributos puede ser a su vez un objeto podría darse la siguiente situación:

1. Nombre de la clase
    a. Nombre de la clase del primer atributo
    b. Datos del primer atributo
    c. ...
    d. Datos del último atributo.
2. ...
3. Datos del último atributo.

#### 7.4.5.2 Algoritmo de serialización y deserialización

A continuación será explicado el algoritmo de serialización y deserealización.

Para los objetos serializables el método writeObject permite a una clase controlar la serialización de sus propios atributos. Cada subclase de un objeto serializable debe definir su propio método writeObject.

El método writeObject es el responsable de salvar el estado de los objetos de esta clase. El formato y estructura es responsabilidad completa de la clase.

De la misma manera, el método readObject permite la deserialización de sus propios atributos y es el responsable de recuperar el estado del objeto.

#### 7.4.5.3 Serialización

La clase ObjOutputStream es la encargada de la serialización. Cuando una instancia de esta clase es creada, al cual se le es asociado un flujo de bytes de escritura, se puede comenzar con el proceso de serialización.

Cuando la clase ObjOutputStream recibe una llamada al método writeObject obtiene como parámetro el objeto a serializar.

En primer lugar se debe obtener el nombre de la clase del objeto a serializar a través de una llamada al método object.getClass().getName() heredado desde la clase Object. Luego se escribe el nombre de la clase en el flujo de bytes de escritura y se llama al método writeObject del objeto serializable recibido como parámetro para que se lleve a cabo la serialización de sus atributos.

Cabe destacar que es un proceso recursivo, por lo que no hay problemas para serializar objetos anidados, ya que son las propias llamadas a sucesivos métodos writeObject las que se encargan de realizar dicha recursividad.

### 7.4.5.4 Deserealización

Este proceso está profundamente relacionado con el de serialización, ya que dependiendo del orden que éste haya seguido habrá que leer una u otra cosa.

A partir del flujo de bytes que contiene uno o más objetos serializados se debe leer el nombre de la clase serializada. Posteriormente se debe crear una instancia de esta clase mediante el método:

*object = (Serializable)Class.forName(className).newInstance().*

Luego, se debe llamar al método readObject para que los valores de sus atributos sean cargados, leyendo los diferentes valores de los atributos que componen el objeto en el orden en que fueron escritos. Este orden es conocido por la clase dado que el mismo anteriormente los escribió al serializarce.

### 7.4.5.5 Interfaces y clases participantes

La única interfaz necesaria para proveer el mecanismo de serialización en J2ME es la siguiente:

- **Serializable**: sirve para establecer que una clase es serializable, ya que al implementarla se deberán definir los métodos necesarios para poder escribir una instancia de la clase en el flujo de bytes con un formato de dato y posteriormente recuperarse desde dicho flujo de datos.

Las dos principales clases necesarias para poder llevar a cabo el proceso de serialización son:

- **ObjInputStream**: representa un objeto serializado del cual se van a poder leer los diferentes objetos que éste alberga a través de su método readObject. Esta clase esta asociada a un flujo de bytes, que es realmente la información, y hereda de la clase DataInputStream, por lo que todos sus métodos estarán disponibles en esta clase.

- **ObjOutputStream**: se encarga de serializar los objetos recibidos en su método *writeObject* en el flujo de bytes obtenido en su constructor. Esta clase se apoya en los métodos heredados desde su superclase DataOutputStream. Los diferentes objetos pasados como parámetro al método writeObject se convertirán en un flujo lineal de bytes con una estructura predefinida que permitirá posteriormente reconstruirlos a través de la clase ObjInputStream.

### 7.4.5.6 Ejemplo de Serialización

En este ejemplo se mostrará como serializar una colección de datos, creando una clase VectorS que representara un vector serializable. Esta clase implementará la interfaz Serializable, definiendo la implementación de los métodos readObject y writeObject, y extenderá la clase Vector para heredar todas sus operaciones.

Antes de mostrar el código de esta clase se expondrán los métodos necesarios para la interfaz Serializable y las clases ObjInputStream y ObjOutputStream.

```
public interface Serializable {
        public void readObject(ObjInputStream in) throws IOException;
        public void writeObject(ObjOutputStream out) throws IOException;
}

public class ObjInputStream extends DataInputStream {

        public ObjInputStream(DataInputStream in) throws IOException{
                super(in);
                this.in = in;
        }
        public synchronized Object readObject() throws IOException, InstantiationException,
                                                        IllegalAccessException, classNotFoundException{
                        String className =((DataInputStream)in).readUTF();
                        className.trim(); //limpia los espacios en blanco de stream
                        Serializable object = (Serializable)Class.forName(className).newInstance();
                        object.readObject(this);
                        return object;
        }
}

public class ObjOutputStream extends DataOutputStream {

        public ObjOutputStream(DataOutputStream out) {
                super(out);
                this.out = out;
        }
        public void writeObject(Serializable o) throws IOException{
                ((DataOutputStream)out).writeUTF(o.getClass().getName());
                o.writeObject(this);
        }
}
```

A continuación se muestra la implementación de la clase VectorS, donde se supone que los elementos contenidos en la colección también son objetos serializables.

```
public class VectorS extends Vector implements Serializable{

        public void readObject(ObjInputStream in) throws IOException {
                int cant =in.readInt(); //escribe la cantidad de elementos que posee el vector
                try{
                for (int i = 0; i < cant; i++) { // escribe cada elemento en el stream de datos
                        String className =in.readUTF();
                        Object object = (Object)Class.forName(className).newInstance();
                        ((Serializable)object).readObject(in);
                        this.addElement(object);
                }
                }catch (Exception e) {
                        e.printStackTrace();//imprime el error ocurrido
                }
        }

        public void writeObject(ObjOutputStream out) throws IOException {
                out.writeInt(this.size()); // lee la cantidad de elementos del vector que debe leer
                Enumeration iter = this.elements();
                while (iter.hasMoreElements()) { //recupera cada elemento desde el stream de datos
                        Object element = (Object) iter.nextElement();
                        out.writeObject((Serializable) element);
                }
        }
}
```

Un ejemplo de la recuperación de un VectorS enviado a través de la red como un Stream de datos es:

```
//se conecta con la URL pasando los parámetros necesarios
HttpConnection  conn = (HttpConnection)Connector.open(url + reqParams);
//Abre el DataInputStream y lee los bytes desde el servidor
DataInputStream dIn  dIn = conn.openDataInputStream();
try{
        ObjInputStream  sal = new ObjInputStream(dIn);
        VectorS leido = (VectorS) sal.readObject();
}catch(IOException e){
System.out.println(e.getMessage());
}
```

En el caso de encontrarnos en un Servlet y escribir los datos desde el Stream provisto por la respuesta del mismo, los pasos necesarios para escribir el VectorS a enviar son:

```
VectorS resultado = new VectorS();
//se le agregan los objetos serializables que se quieran enviar
....
DataOutputStream datos = new DataOutputStream(response.getOutputStream());
 // Siendo response una instancia de la clase HttpServletResponse.

ObjOutputStream out = new ObjOutputStream(datos);
out.writeObject(resultado);
out.flush();
out.close();
```

# 8 Conclusiones y trabajos futuros

## 8.1 Conclusiones

En este trabajo se han presentado los requerimientos para la implementación de una arquitectura que soporte la localización automática de vehículos permitiendo también la adaptación al contexto para usuarios móviles. También se han establecido un conjunto de operaciones y propiedades de contexto que han de ser tenidas en cuenta para este tipo de aplicaciones. Una de las características más importantes de la arquitectura propuesta es que soporta la evolución de futuras operaciones o propiedades de contexto; la importancia de esta característica está basada en el dinamismo que se presenta cuando se manipulan, como parte del sistema, los contextos de usuario.

A su vez, en este trabajo, fue desarrollado un prototipo basado en la arquitectura propuesta, el cual implementa solo una porción reducida de todos los aspectos que pueden ser tomados en cuenta.

Algunas conclusiones interesantes que se pueden destacar del trabajo realizado son:

- *Combinación de áreas de investigación*: en este trabajo se logro relacionar cuatro tópicos diferentes (Mobile Computing, AVL, GIS y Context-Aware) de manera natural, permitiendo a un sistema de localización automática de vehículos manipular usuarios móviles distribuidos en diferentes lugares del mapa con el que se está trabajando, y adaptar la información suministrada a estos usuarios móviles dependiendo del contexto en que se encuentran.

- *Extensibilidad del modelo:* como ya ha sido expresado, seguramente existen otras operaciones o contextos que no han sido presentadas en este trabajo o que pueden variar a lo largo del tiempo. Gracias a la flexibilidad conseguida en la arquitectura de objetos es posible agregar fácilmente cualquier operación o propiedad de contexto que pueda llegar a necesitarse.
  Dichas extensiones se realizan básicamente sub-clasificando y redefiniendo métodos abstractos o comunes de toda la jerarquía. Se debe destacar que dicha flexibilidad se debe en gran medida a la utilización de los patrones de diseño [Gamma, 95], los cuales brindan soluciones acertadas a problemas recurrentes.

- *Aspectos de implementación:* la implementación del prototipo fue realizada usando dos protocolos diferentes del lenguaje de programación Java, J2SE para la implementación de una central y atención de requerimientos HTTP y, J2ME para la implementación de los usuarios móviles. A su vez, se utilizo la suite de componentes MapObject-java para la visualización de mapas digitales y las operaciones sobre estos. De las implementaciones realizadas se puede con concluir:

  - Java permite expresar perfectamente conceptos importantes para la teoría de O.O., como pueden ser interfaces, clases y métodos abstractos. Este poder expresivo permite obtener software de gran calidad.

- J2SE permitió implementar todas las operaciones necesarias sin mayores complicaciones, dado que en la actualidad existen paquetes (packages) adicionales que implementan soporte para diversos fines.

- Por el contrario, J2ME aún sigue siendo limitado en algunos aspectos, por lo cual, implementar ciertas funcionalidad (como por ejemplo comunicación por sockets, serialización, etc.) requirió una investigación muy profunda del paradigma para lograr realizarlas.

- Por otro lado, la utilización de MapObject-Java para operar con mapas digitales fue sencilla y apropiada a los fines del prototipo.

## 8.2 Trabajos futuros

El alcance de Context-Aware (Sensibilidad al contexto) es muy amplio y obviamente no fueron abarcados todos los problemas existentes en éste área en el marco de este trabajo. A continuación se mencionaran algunos de ellos que merecen una posterior investigación y desarrollo.

- *Modelo para la toma de información de contexto*

  Aunque el desarrollo de un modelo confiable y escalable encargado de la toma de datos que componen la información de contexto esta fuera del alcance de este trabajo, es un tema de vital importancia para Context-Aware, dado que sin la información apropiada no se pueden realizar las adaptaciones adecuadas.
  Si bien capturar la información parece un tema menor, hay varios trabajos que demuestran la complejidad del mismo, tales como "Context Toolkit Approach" [Dey, 99], "Hydrogen Approach" [Schwinger, 02], etc.

- *Configuración de acciones dados los contextos para los usuarios*

  Las adaptaciones de información requeridas pueden cambiar rápidamente para los diferentes requerimientos realizados por los usuarios, así como también los usuarios podrían querer realizar cada uno sus propias reglas de adaptación.
  La posibilidad de que un usuario pueda realizar estas modificaciones sin tener que modificar el código, incorporar nuevas clases, o cualquier otro cambio necesario, facilitaría la incorporación y modificación de las adaptaciones realizadas. Pero incorporar esta posibilidad trae aparejado nuevos problemas, por ejemplo: ¿Cómo manejar estos cambios mientras que el sistema es mantenible, reusable y extensible eficientemente? ¿Cómo modelar y manejar (representar) reglas, para una mayor reutilización, capacidad de mantenimiento y funcionamiento?
  Existen varios trabajos que investigaron este tema, entre los trabajos destacados se encuentra el desarrollado por Arsanjani [Arsanjani, 01], el cual propone algunos patrones a tener en cuenta a la hora de trabajar con reglas de adaptación.

Existen otros temas, no relacionados al contexto, que valen la pena incorporar a este trabajo en una siguiente etapa. Algunos de ellos son:

- Conexión a distintos servidores para la toma de datos cartográficos

  En este trabajo solo se interactúa con un servidor central en el cual se encuentra la totalidad de la cartografía con la que se esta trabajando, pero seria ventajoso tener en cuenta que si la empresa aseguradora estuviera distribuida en diferentes localidades y los usuarios móviles se desplazaran entre estas, se pudiesen comunicar automáticamente con el servidor correspondiente a la ubicación del usuario móvil, el cual posee la cartografía apropiada a su posición. Además, debería tenerse en cuenta que al trabajar con varios servidores podría ocurrir que un vehiculo en movimiento realice una solicitud de datos en un determinado lugar pero, al irse desplazando, cuando se recibe la respuesta esta no pueda ser brindada por el servidor al que se le realizo la solicitud, ya que la zona en la que se encuentra el vehiculo actualmente no es la correspondiente a éste servidor, o esta información recibida ya no sea adecuada dado que el vehiculo se encuentra en una nueva zona.

- Persistencia

  Este trabajo no contempla la persistencia de los sucesos ocurridos en el sistema, sino que todo se encuentra en memoria. Si los datos fueran persistidos podrían realizarse estadísticas o incluso Data Mining[7]. Por ejemplo, al conocer lugares y horarios de circulación normal del vehículo se podría sospechar posibles robos cuando no estuviese en dichas situaciones normales.
  Existen numerosas alternativas para realizar la persistencia y seria de importancia evaluarlas encontrando la más adecuada de manera de lograr una capa de persistencia independiente del modelo, el cual no debería verse afectado.

[7] Data Mining es un proceso de descubrimiento de información en bases de datos

# 9 Referencias

**[Alvarion]** Breezecom Breezenet Product Homepage. http://www.alvarion.com/

**[Aronoff, 89]** Geographic information systems: A management perspective. WDC Publications, Ottawa.

**[Arsanjani, 01]** Ali Arsanjani, "Rule Object 2001: A Pattern Language for Adaptive and Scalable Business Rule Construction". PLoP 2001.

**[Bauer, 03]** Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic. Diplomarbeit.

**[Bluetooth]** The official Bluethoth Website. http://www.bluetooth.com/

**[Brown, 96]** Brown, P.J., "The Stick-e Document: A Framework For Creating Context-aware Applications", In the Proceedings of the Electronic Publishing, paginas 259-272, Laxenburg, Austria, IFIP. Septiembre 1996.
http://www.google.com.ar/search?q=cache:DZWbf5-cidUJ:cajun.cs.nott.ac.uk/compsci/epo/papers/volume8/issue2/2point1.pdf+The+Stick-e+Document:+A+Framework+For+Creating+Context-aware+Applications&hl=es&ie=UTF-8

**[Cheverst, 99]** Cheverst, K., Mitchell, K., and Davies, N. Design of an object model for a context sensitive tourist GUIDE. Computers and Graphics 23, 6 (1999), 883–891.

**[Chtcherbina, 03]** Peer-to-peer coordination framework (p2pc): Enabler of mobile ad-hoc networking for medicine, business, and entertainment. In Proceedings of International Conference on Advances in Infrastructure forElectronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (L'Aquila/Italy, January 2003).

**[Davies, 98]** Davies, N., Mitchell, K., Cheverst, K. and Blair, G.S., "Developing a Context Sensitive Tourist Guide", Proc First Workshop on Human Computer Interaction for Mobile Devices, Glasgow. March 1998.
http://citeseer.nj.nec.com/cache/papers/cs/2480/ftp:zSzzSzftp.comp.lancs.ac.ukzSzpubzSzmpgzSzMPG-98-24.pdf/davies98developing.pdf

**[Dey, 99]** Dey, A.K., Salber, D., Abowd, G.D., "A Context-based Infrastructure for Smart Environments", In the Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), pp. 114-128, Dublin, Ireland, Springer Verlag. December 13-14, 1999.

**[Dey, 00]** Dey, A.K., and Abowd, G.D., "Towards A Better Understanding of Context and Context-Awareness. In the Workshop on the What, Who, Where, When and How of Context-Awareness", affiliated with the 2000 ACM Conference on Human Factors in Computer Systems (CHI 2000), The Hague, Netherlands. 1-6 Abril del 2000.

**[Dix, 99]** Dix, A., Rodden, T., Davies, N., Trevor, J., Friday A. and Palfreyman, P.,

"Exploiting Space and Location as a Design Framework for Interactive Mobile Systems", ACM Transactions on Computer-Human Interaction (TOCHI). 1999. http://www.hiraeth.com/alan/papers/context99/context99.pdf

**[Forman, 94]** George H. Forman, John Zahorjan, "The challenges de Mobile Computing", Computer Science & Engineering. March 9, 1994. http://www.csd.uwo.ca/courses/CS848a/Papers/forman94.pdf

**[FranklinFlaschbart, 98]** Franklin and Flaschbart (1998). All gadget and no representation makes jack a dull environment. In the *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments (AAAI Technical Report SS-98-02)*, Palo Alto, CA, AAAI Press. March 23-25, 1998. http://dent.infolab.nwu.edu/infolab/downloads/papers/paper10072.pdf

**[Gamma, 95]** Gamma, E.; Helm, R.; Johnson, R.; and Vlissides, J. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Reading, Addison-Wesley Professional Computing Series.

**[Gordillo, 99]** Developing GIS Applications with Objects. A design Patterns Approach. Silvia Gordillo, Federico Balaguer, Catalina Alba Mostaccio, Fernando Das Neves. "GeoInformática (1999) Kluwer Academic Publishers.", ISSN 1384-6175, Volume vol 3 numero 1 (7 - 32), 1/1/1999.

**[Gordillo, 00]** Sistemas de Referencias para Aplicaciones SIG´s. "VI Congreso Argentino de Ciencias de la Computación CACIC 2000. Ushuaia, Argentina.", 1/10/2000.

**[GSM, 03]** GSM Association Classifications. Location Bases Services. Junuary 2003. http://www.gsmworld.com/documents/lbs/se23.pdf

**[Held, 02]** Modeling of context information for pervasive computing applications. In Proceedings of SCI 2002/ISAS 2002.

**[Hull, 97]** Hull, Richard, Philip Neaves and James Bedford-Roberts (1997). Towards situated computing. In the *Proceedings of the 1st International Symposium on Wearable Computers (ISWC'97)*, pp. 146-153, Cambridge, MA, IEEE. October 13-14, 1997. http://fog.hpl.external.hp.com/techreports/97/HPL-97-66.pdf

**[Ibutton]** Dallas Semiconductors Weather Instruments Product Homepage. http://www.ibutton.com/weather/index.html

**[Indulska, 03]** Experiences in using cc/pp in context-aware systems. In LNCS 2574: Proceedings of the 4th International Conference on Mobile Data Management (MDM2003)

**[Intel]** Intel Network Connectivity http://www.intel.com/network/connectivity/index.htm

**[Johnson, 97]** R. Johnson, B. Woolf. The Type Object Pattern.

http://www.ksc.com/article3.htm

**[Kappel, 02]** Towards a Generic Customisation Model for Ubiquitous Web Applications.

**[Lamming, 94]** Lamming, M. and Flynn, M., "Forget-me-not: Intimate computing in support of human memory", In the Proceedings of the FRIEND 21: International Symposium on Next Generation Human Interfaces, pp. 125-128, Meguro Gajoen, Japan. 1994.
http://www.lamming.com/mik/Papers/fmn.pdf

**[Longley, 02]** P. Longley, M. Goodchild, D. Maguire, D. Rhind. Geographic information, Systems and Science

**[McCarthy, 93]** MCCARTHY, J. Notes on formalizing contexts. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (San Mateo, California, 1993),

**[MIDP 1.0]** JSR-000037 Mobile Information Device Profile (MIDP)
http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html

**[MIDP 2.0]** JSR-000118 Mobile Information Device Profile 2.0
http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html

**[MVC]** http://c2.com/cgi/wiki?ModelViewController

**[Nokia]** Nokia Phones.
http://www.nokia.com/nokia/0,1522,,00.html?orig=/phones/9210/index.html

**[Nokia, 1]** Nokia Wireless Accelerator. http://www.nokia.com/nokia/0,,48276,00.html

**[Otztürk, 97]** A. Towards a model of context for case-based diagnostic problem solving. In Context-97; Proceedings of the interdisciplinary conference on modeling and using context.

**[Pascoe, 98]** Ryan, Nick, Jason Pascoe and David Morse (1998). *Enhanced reality fieldwork: the context-aware archaeological assistant.* Computer Applications and Quantitative Methods in Archaeology. V. Gaffney, M. van Leusen and S. Exxon, Editors. Oxford.
http://www.cs.ukc.ac.uk/research/infosys/mobicomp/Fieldwork/Papers/CAA97/ERFldwk.html

**[Pascoe, 98a]** Pascoe, J., "Adding Generic Contextual Capabilities to Wearable Computers", In the Proceedings of the 2nd IEEE International Symposium on Wearable Computers (ISWC'98), pp. 92-99, Pittsburgh, PA, IEEE. October 19-20, 1998.
http://citeseer.nj.nec.com/cache/papers/cs/21917/http:zSzzSzwww-anw.cs.umass.eduzSzwearableszSzreadingzSzpaperszSzpascoe.98.pdf/pascoe98adding.pdf

**[Pointsix]** Point Six Inc. Homepage.

http://www.pointsix.com/cgi-bin/PointSix.cgi?Point_Sensors

**[Rhind, 90]** Global databases and geographic information systems. In The Association for Geographic Information Yearbook 1990. eds. MJ Foster & PJ Shand, pp. 218-223. Taylor & Francis and Miles Arnold, London.
http://www.dwaf.gov.za/sfra/SEA/GIS/gis_define.htm

**[Rumbough, 98]** The Object Constraint Language (OCL) of UML.
http://www-st.inf.tu-dresden.de/fs/lectnotes/fss5a.pdf

**[Ryan, 98]** Ryan, Nick, Jason Pascoe and David Morse (1998). *Enhanced reality fieldwork: the context-aware archaeological assistant*. Computer Applications and Quantitative Methods in Archaeology. V. Gaffney, M. van Leusen and S. Exxon, Editors. Oxford.
http://www.cs.ukc.ac.uk/research/infosys/mobicomp/Fieldwork/Papers/CAA97/ERFldwk.html

**[Schilit, 94]** Schilit, B. N., Adams, N. I. and Want, R., "Context-aware Computing Applications", In the Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, pp. 85-90, Santa Cruz, CA, IEEE. 8-9 de diciembre de 1994.
http://seattleweb.intel-research.net/people/schilit/wmc-94-schilit.pdf

**[SchilitTheimer, 94]** Schilit, B. N. and Theimer, M., "Disseminating Active Map Information to Mobile Hosts", IEEE Networks, paginas 22-32, Octubre 1994.

**[Schmidt, 98]** Schmidt, A., Beigl, M. and Gellersen, H. W., "There is More to Context than Location", In Interactive Applications of Mobile Computing, Rostock, Germany, 24-25, November 1998.
http://www.comp.lancs.ac.uk/~albrecht/pubs/pdf/schmidt_cug_elsevier_12-1999-context-is-more-than-location.pdf

**[Schwinger, 02]** T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, Context-Awareness on Mobile Devices - the Hydrogen Approach, Software Competence Center Hagenberg. 2002 IEEE.

**[SDNS, 04]** The Source for Developers, a Sun Developer Network Site. J2ME and Location-Based Services. March 2004.
http://developers.sun.com/techtopics/mobility/apis/articles/location/

**[Sony]** Sony Electronics Viao Homepage. http://www.sonystyle.com/vaio/

**[Struts]** http://struts.apache.org/

**[SWING]** http://java.sun.com/products/jfc/index.jsp

**[Trimble]** Trimble Navigation Limited. Todo sobre GPS. http://www.trimble.com/gps/

**[UAProf]** WAPFORUM. User Agent Profile (UAProf).
http://www.wapforum.org

**[ValuedLBS, 04]** Valued LBS research. Location Based Services. Updated May 26, 2004.
http://www.lboro.ac.uk/research/esri/lbs/

**[Wereable Computing]** Wereable Computing. http://www.media.mit.edu/wearables/

# Apendice I J2ME

En las siguientes secciones se describirá brevemente en que consiste la plataforma J2ME, las partes que la componen y sus aspectos más destacables.

## *I.I Definiendo J2ME*

Dada la aparición de pequeños dispositivos surgió la necesidad de aplicaciones en los mismos. Con el fin de solucionar esta creciente demanda Sun introdujo la plataforma Java 2 Micro Edition (J2ME). Esta nueva plataforma esta orientada a pequeños dispositivos tales como teléfonos móviles y PDAs, con pantallas reducidas y capacidades de memoria y cálculo limitadas.

Antiguamente estos dispositivos venían con un software empotrado al cual era prácticamente imposible añadir nuevas funcionalidades. Con la tecnología Java es posible la distribución dinámica y segura de diferentes aplicaciones a través de cualquier tipo de red.

Sun ha agrupado su tecnología Java en tres ediciones, cada una destinada a un área tecnológica diferente:

- **Java 2 Enterprise Edition** (J2EE): orientada a empresas que necesitan proporcionar servicios a clientes y proveedores a través de soluciones e-commerce y e- business.
- **Java 2 Standart Edition** (J2SE): para el mercado de computadoras personales (aplicaciones de usuario, applets, etc).
- **Java 2 Micro Edition** (J2ME): combina las necesidades de los fabricantes de dispositivos embebidos, los proveedores de servicios que desean distribuir información a sus clientes a través de estos dispositivos y en tercer lugar, a los creadores de contenidos para que estén disponibles así mismo en estos dispositivos.

La figura I.1 da una visión esquemática del alcance de cada una de las tecnologías java.

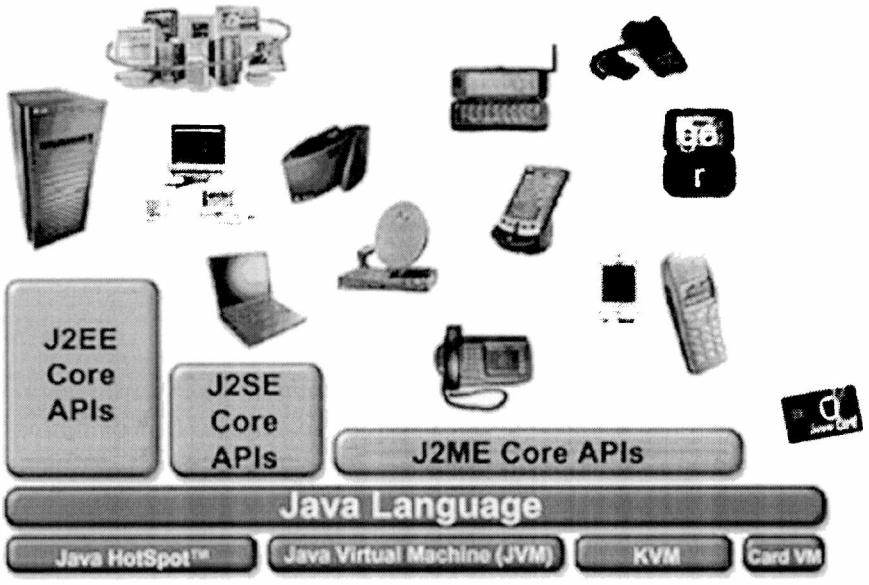


**Figura I.1: Tecnologías Java**

Cada una de las ediciones difiere respecto al resto en la maquina virtual que poseen y en las APIs que utilizan, pero siempre tienen en común el lenguaje de programación que emplean, Java.

## *I.II Otras tecnologías inalámbricas*

Aquí se detallaran algunas de las tecnologías y servicios inalámbricos con el fin de esclarecer sus diferencias con J2ME, nombrando las más conocidas.

- **WAP**: acrónimo de Wireless Application Protocol, es una tecnología introducida en el mercado en 1995 que permite a los dispositivos inalámbricos recibir datos de Internet y mostrarlos en las pantallas de los mismos. Se piensa en WAP como una tecnología que da soporte a buscadores Web en móviles, pero en realidad no es una aplicación sino un protocolo.
- **SMS**: Short Messaging Service, es una tecnología que da soporte al envío y recepción de mensajes cortos de texto en dispositivos móviles. Otra característica interesante de SMS es que emplea una mensajería unificada, lo que permite acceder a mensajes de voz, correo electrónico y faxes desde un dispositivo móvil.
- **I-MODE**: fue introducido al mercado por la compañía Japonesa NTT DoCoMo, esta tecnología compite con WAP, ya que de igual modo ofrece un mecanismo de acceso a Internet a través de dispositivos móviles. I-MODE dispone de un lenguaje de tags similar a HTML denominado compact HTML (cHTML). La mayoría de sus usuarios se encuentran en Japón y otros países asiáticos.

La principal diferencia con estas tecnologías es que J2ME permite el desarrollo de aplicaciones genéricas que podrán ejecutarse en un determinado tipo de dispositivos, mientras que WAP e I-MODE simplemente proporcionan acceso a Internet desde un dispositivo móvil.

## I.III Arquitectura de J2ME

Con el fin de conseguir una mayor flexibilidad y adaptación a diversos tipos de dispositivos, J2ME se estructura en tres niveles.

- **Maquina virtual**

  Adaptada a dispositivos con capacidades limitadas, esta máquina esta ligada a una configuración. En la actualidad existen dos tipos de máquinas virtuales en J2ME: la CVM (C Virtual Machine) y la KVM (Kilo Virtual Machine).

  La CVM esta orientada a dispositivos embebidos y electrónica de consumo (dispositivos como TV digital, electrodomésticos, etc.) ligada a la configuración CDC, requiere mayores recursos que su hermana pequeña, la KVM.

  La KVM tiene sus orígenes en Spotless (máquina virtual para el sistema operativo PalmOS). Diseñada desde cero para dispositivos con poca memoria, capacidad de proceso limitada, limitaciones de batería y conexiones a red intermitentes, esta máquina va unida a la configuración CLDC.

- **Configuración**

  Una configuración es el conjunto mínimo de clases disponibles (o características) en una categoría de dispositivos. Las categorías se establecen según requisitos similares de memoria y procesamiento. A su vez, cada configuración esta profundamente relacionada a una maquina virtual.

  Actualmente existen dos configuraciones en J2ME:

  - *Connected Device Configuration* (CDC)

    Esta configuración esta orientada a dispositivos con 2 MB o más de memoria total, incluyendo RAM y ROM, conexión a red fija, soporte completo de la especificación de la JVM e interfaz de usuario relativamente limitada.

  - *Connected, Limited Device Configuration (CLDC)*

    Los requisitos de esta configuración son notablemente inferiores a la CDC. La memoria disponible requerida para la plataforma Java varía entre los 160 y 512 KB y los procesadores requeridos son de 16 o 32 bits. Otros requerimientos son la conectividad a algún tipo de red, a menudo inalámbrica, con conexión intermitente y ancho de banda limitado; y bajo consumo de energía.

- **Perfil (Profile)**

  El perfil es un conjunto de clases Java que complementan una configuración para un conjunto específico de dispositivos.

  Los perfiles permiten la portabilidad de aplicaciones J2ME entre dispositivos diferentes.

  Actualmente existe una implementación sobre CLDC, la MIDP y una especificación, la del PDA Profile.

  La figura I.2 muestra la arquitectura de J2ME.

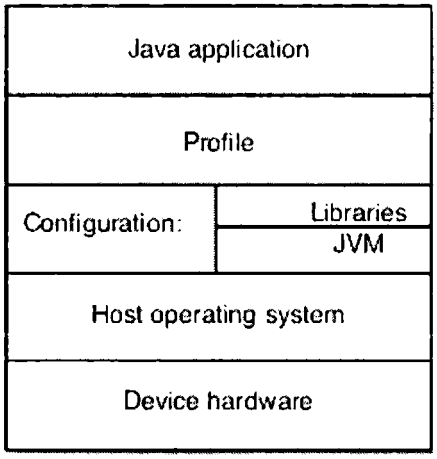

**Figura I.2: Arquitectura J2ME.**

La plataforma J2ME consiste de un conjunto de capas que soportan un entorno de ejecución básico con corazón en las librerías Java y una maquina virtual (VM) , un conjunto de interfaces de programación de aplicaciones a nivel sistema (APIs) en una configuración, y un conjunto de APIs a nivel de aplicación en un perfil.

Debido a que se trabajó con dispositivos móviles de poca capacidad, en las secciones siguientes se explica más detalladamente la KVM, el CLDC y el MIDP.

## I.III.i Máquina Virtual K (KVM)

La Kilo Virtual Machine es una máquina virtual Java altamente portable y compacta, pensada y diseñada para funcionar en dispositivos con recursos limitados y conexión a red intermitente.

El objetivo de la tecnología KVM es crear una JVM lo más pequeña y completa posible, que mantenga los aspectos más importantes del lenguaje de programación Java y que funcione en dispositivos con procesadores de 16 o 32 bits y con unos pocos cientos de KiloBytes (originalmente 128 Kbytes) de memoria.

## I.III.ii Configuración CLDC

Este nivel es menos visible para los usuarios, pero es muy importante para los implementadores de perfiles. Define el conjunto mínimo de características de la máquina virtual y las librerías de clases disponibles en una determinada categoría de dispositivos. Representaría una franja horizontal de dispositivos con un mínimo denominador común, que los desarrolladores van a poder asumir que tienen todos los dispositivos con esa configuración.

Las configuraciones son especificadas a través de la iniciativa Java Community Process y sus implementaciones testeadas a través del test de compatibilidad TCK (compatibility test kit).

### I.III.iii Perfil MIDP

Este nivel es más visible a los usuarios y desarrolladores de aplicaciones. Las aplicaciones se desarrollan sobre un determinado perfil que a su vez esta implementado sobre una determinada configuración.

El perfil define un conjunto de APIs y características comunes para una franja vertical de dispositivos. Las clases de un perfil permiten el acceso a funcionalidades específicas de los dispositivos como la interfaz gráfica, funcionalidades de red, almacenamiento persistente, etc.

Las aplicaciones desarrolladas sobre un determinado perfil van a ser portables a cualquier dispositivo que soporte ese perfil. Cabe destacar que un dispositivo puede soportar varios perfiles y que sobre una configuración pueden existir diversos perfiles.

## *I.IV J2ME vs. J2SE*

J2ME ha sido creado para adaptarse a las características de los nuevos dispositivos inalámbricos, tales como teléfonos móviles y PDAs. Consecuentemente existirán diferencias con la versión estándar de Java destinada a PCs. Algunas de las principales diferencias son las siguientes:

1 **Tipos de datos**

J2ME soporta un subconjunto de los tipos de datos de J2SE. Los tipos **float** y **double** no están soportados por dos razones: la mayoría de los dispositivos CLDC no disponen del hardware de unidad de coma flotante y en segundo lugar, es una operación muy costosa de llevar a cabo.

2 **Preverificación** (on-line y off-line)

Al contrario que en J2SE, en donde la verificación del código se realiza en tiempo de ejecución, en J2ME parte de la verificación se realiza off-line, es decir, fuera del dispositivo. Esto tiene la finalidad de reducir la carga de la máquina, llevando a cabo el resto de la verificación on-line.

3 **Descriptor y manifiesto**

Al empaquetar clases en J2ME es necesaria la inclusión de dos archivos especiales que contienen información de las aplicaciones que se incluyen, estos archivos se denominan descriptor (.jad) y el manifiesto (manifest.mf).

4 **Librería gráfica**

J2ME define un nuevo conjunto de clases para la creación de las interfaces gráficas. Estas clases están adaptadas a dispositivos con memorias muy limitadas y pantallas de tamaño reducido, y reducidas a solo un par de posibilidades. La librería grafica AWT desaparece en esta nueva plataforma.

5 **Desaparición del main**

Al contrario de las aplicaciones de J2SE en J2ME no existe una función main como punto de entrada para la ejecución del programa.

6 **Ausencia del recolector de basura** (Garbage Collector)

Con el objetivo de reducir la utilización de memoria, desaparece el recolector de basura en J2ME siendo necesario eliminar de manera explícita todos aquellos elementos que no vayan a ser utilizados nuevamente.

7 **Limitación en el tratamiento de errores**

Aunque J2ME soporta el tratamiento de excepciones, el conjunto de excepciones incluidas en las librerías de la configuración se ha reducido sustancialmente. Esto se debe a dos razones:

- En los dispositivos embebidos la recuperación de errores es algo generalmente específico de cada uno de ellos. Algunos dispositivos tratan de recuperarse de estas situaciones, mientras que otros se resetean al encontrar un error. El programador no puede tener en cuenta este tipo de situaciones ya que es propio de cada dispositivo.
- La implementación de las capacidades de manejo de excepciones de acuerdo a la especificación del lenguaje Java es relativamente costoso, y supondría una carga significativa en la implementación dadas las estrictas limitaciones de memoria en los dispositivos CLDC.

8 **Otras características han sido eliminadas de la maquina virtual Java** que soporta el CLDC debido a la carencia de ciertas librerías o con el objetivo de mantener el modelo de seguridad establecido por el CLDC. Entre las características eliminadas cabe mencionar:

- *Interfaz Nativa Java* (JNI)

El modelo de seguridad proporcionado por el CLDC asume que el conjunto de funciones nativas debe ser cerrado (Sandbox Model). Así mismo supone un incremento notable en los requerimientos de hardware.

- *Cargadores de clases definidos por el usuario*

Al igual que con JNI el hecho de poder reconfigurar o crearse nuevos cargadores es parte de las restricciones del modelo de seguridad establecido.

- *Reflexión*

Con el objetivo de mantener la seguridad así como reducir el tamaño de la maquina virtual la reflexión no es soportada.

- *Grupos de hilos e hilos demonio*

Aunque el CLDC implementa los multihilos no existe soporte para grupo de hilos e hilos demonios (daemons threads). Las operaciones

como arranque y parada de los threads se pueden aplicar de manera individual. En caso de que los desarrolladores de aplicaciones quieren utilizar estas características deberán implementarlas.

- *Finalización*

Las librerías CLDC no incluyen el método Object.finalize() con lo que la finalización de instancias de clases no es soportada.

Se puede concluir que estas características han sido eliminadas principalmente (sin tener en cuenta la carencia de soporte para como flotante debido a la ausencia del soporte de hardware necesario) debido a las estrictas limitaciones de memoria en primer lugar y a los problemas de seguridad que surgen debido a la ausencia de un modelo de seguridad completo como el de la edición J2SE.

## I.V MIDlet

Un MIDlet es la aplicación Java desarrollada de acuerdo a la especificación MIDP. Estas aplicaciones además de ser construidas en base a las restricciones de las APIs CLDC y del MIDP deben pasar por un proceso de preverificación previamente a su distribución.

Al igual que los applets y otras aplicaciones Java, las clases de los MIDlets son almacenadas en archivos bytecode con la extensión .class.

## I.VI MIDletSuite

Se conoce por MIDletSuite a la colección de MIDlets empaquetados conjuntamente en un archivo JAR.

Los MIDlets incluyen dentro del archivo jar los siguientes elementos:

- Las clases Java de cada uno de los MIDlets que incluyen
- Clases compartidas
- Los archivos de recursos utilizados por los MIDlets
- Manifiesto que describe los contenidos del archivo JAR y especifica los atributos utilizados por el software encargado de llevar a cabo la instalación de los MIDlets.

Además del archivo JAR es necesario un descriptor de las aplicaciones (archivo JAD) que contiene un conjunto predefinido de atributos empleados para la recuperación e instalación de los MIDlets. Todos los atributos incluidos en el descriptor son accesibles desde los MIDlets. Cabe mencionar que el programador puede añadir sus propios atributos al MIDlet siempre que estos no empiecen por MIDlet-.

La figura I.3 esquematiza un MIDletSuite.

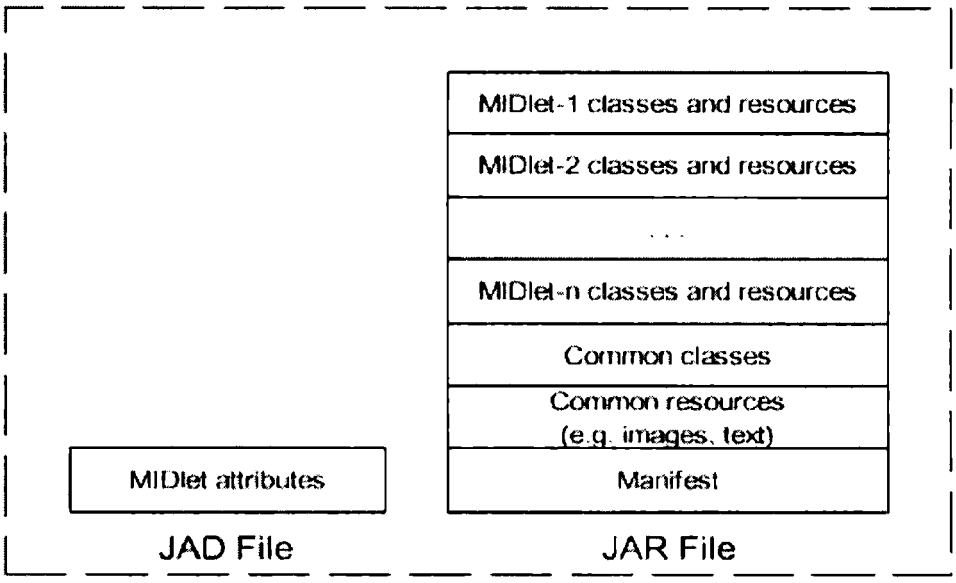


**Figura I.3: MIDletSuite**

La idea del MIDletSuite es la de permitir a todos los MIDlets que contenga compartir los recursos limitados de los dispositivos móviles, siendo inaccesible al resto. Para añadir un nuevo MIDlet al MIDletSuite es necesario volver a generar el .JAR y descargar de nuevo la aplicación.

## *I.VII Comunicación CLDC*

El funcionamiento de red en J2ME tiene que ser muy flexible para soportar una gran variedad de dispositivos, para ello se introduce un nuevo concepto, denominado Marco Genérico de Conexión (MGC), consistente en abstraer el funcionamiento de red y el de los archivos de entrada/salida. La idea consiste en crear todas las conexiones utilizando un método estático (open) de una clase del sistema llamada Connector. Si todo funciona correctamente este método devolverá un objeto que implementa una de las interfaces genéricas de conexión. Existe una jerarquía de estas interfaces siendo la interfaz Connection su raíz.

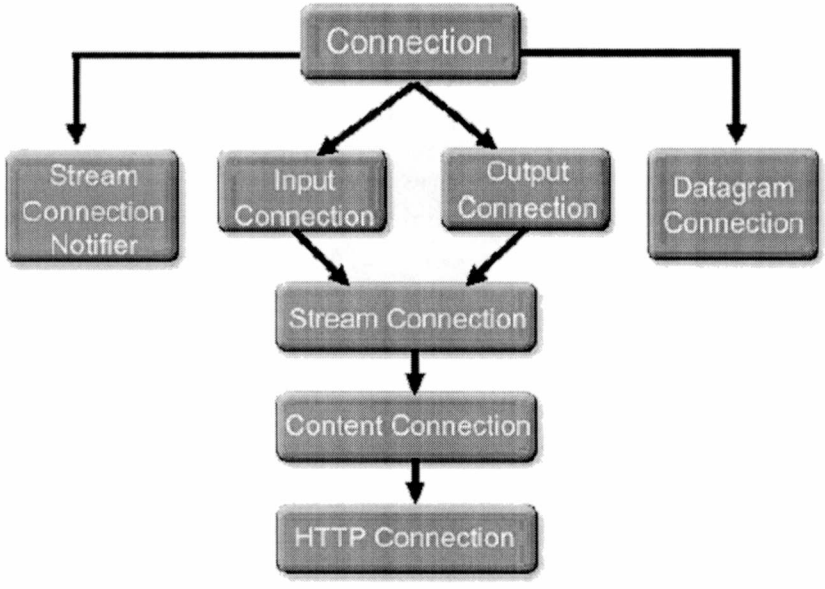


**Figura I.4: Jerarquía de interfaces del MGC.**

Cualquier llamada a este método tendrá la siguiente forma:

Connector.open("<protocol>:<address>;<parameters>");

El objetivo principal de esta abstracción consiste en aislar en la mayor medida posible las diferencias que puedan existir entre las llamadas de cualquier protocolo que se pueda implementar en el perfil.

El MGC incluido en el CLDC no especifica los protocolos soportados ni incluye implementaciones de protocolos concretos (en realidad se incluye alguna implementación a modo de ejemplo, aunque no forma parte de la especificación), si no que es labor del perfil las decisiones de diseño e implementación.

## *I.VIII Interfaz de usuario en MIDP*

Con el fin de permitir flexibilidad y control sobre la interfaz de usuario, el perfil divide las APIs en dos niveles: de bajo y alto nivel.

La API de más alto nivel esta diseñado para aplicaciones en las que se desea un alto grado de portabilidad ente diferentes dispositivos. Para lograr esta portabilidad es necesario llegar a un compromiso entre la cantidad de dispositivos que pueden soportar esta interfaz y el control que los programadores tienen sobre éste.

Por otro lado, la API de bajo nivel tiene un nivel de abstracción menor y requiere ciertas adaptaciones para lograr su adaptabilidad. Esta API está diseñada para aplicaciones que necesitan control sobre los elementos gráficos concretos o eventos de bajo nivel como pueden ser pulsaciones de teclas o botones. Mediante esta API se posee acceso a funcionalidades específicas de los dispositivos. La Figura I.5 muestra la estructura de clases utilizadas para las interfaces en MIDP.

A medida que se utilizan más funcionalidades y clases de la API de bajo nivel la portabilidad va a ser más complicada, ya que con esta API se accede a funcionalidades concretas que puede no tengan algunos dispositivos.

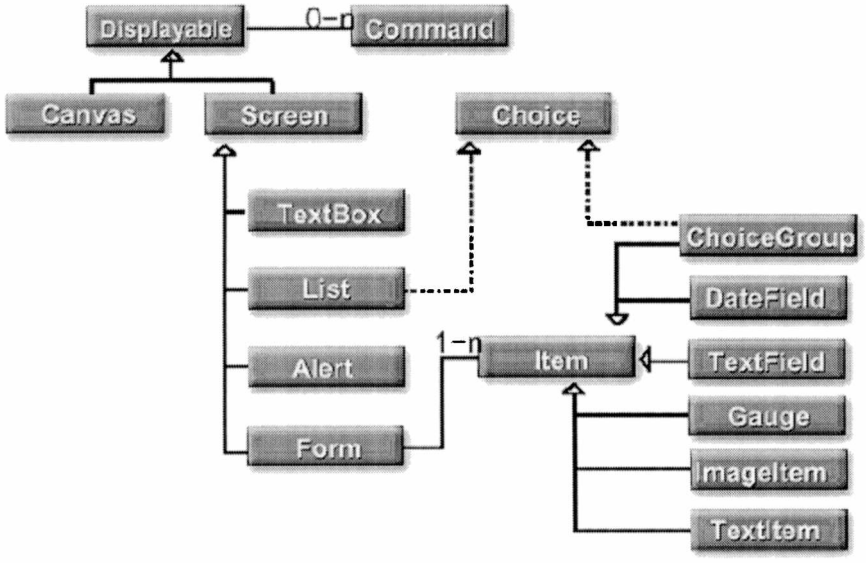


**Figura I.5: Jerarquía de clases de la interfaz de usuario.**

A continuación se hará una breve descripción de las clases más importantes:

- **Displayable**: es un objeto que tiene la capacidad de ser puesto en un Display. Un objeto displayable puede tener comandos y listeners asociados a él. Los contenidos mostrados y su interacción con el usuario son definidos por las subclases.
- **Screen**: es la superclase común de todas las clases de interface de usuario de alto nivel. Agrega título y un "ticket-tape" (porción de texto aclarativo) a la clase Displayable. Los contenidos mostrados y su interacción con el usuario son definidos por las subclases.
- **TextBox**: es una pantalla que permite al usuario ingresar y editar texto.
- **List**: proporciona una lista de opciones. Implementa la interface choice.
- **Alert**: actúa como una pantalla que proporciona información al usuario sobre un error o condición excepcional.
- **Form**: actúa como un contenedor para los otros elementos UI.
- **Choice**: define una selección desde un número predefinido de opciones.
- **Item**: es la superclase de los componentes que pueden ser agregados en un Form o Alert. Todos los objetos Item tienen un Label que es mostrado cerca de las componentes.
- **TextField**: permite al usuario ingresar y editar un texto. Múltiples TextFields pueden ser puestos en un Form.
- **ChoiceGroup**: proporciona un grupo de opciones.
- **DateField**: es un componente editable para presentar información de fecha y hora. Este puede ser puesto en un Form.
- **Gauge**: implementa un gráfico de barras que se pretende usar en un form.

## *I.IX Ciclo de vida de un programa J2ME*

Un programa J2ME es una aplicación de MIDP que hereda de la clase MIDlet.

Un MIDlet tiene uno de 3 estados posibles: activo, pasivo y destruido. Estos se corresponden con los siguientes métodos:

- startApp()
- destroyApp()
- pauseApp()

El administrador del entorno de ejecución de las aplicaciones es el que llama directamente a estos métodos, ya que un MIDlet no cuenta con el método "main".

En la Figura I.6 se muestra el los estados y transiciones posibles de un MIDlet.

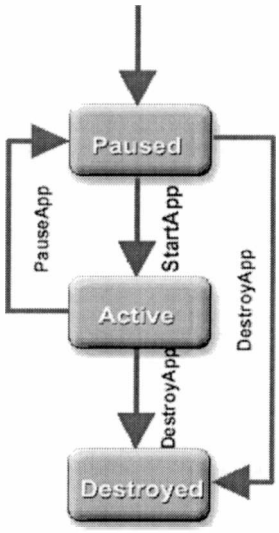


**Figura I.6: Ciclo de vida de un MIDlet.**

El método de esta clase permite al gestor de software de la aplicación crear, iniciar, pasar al estado de pausa y destruir el MIDlet.

Un MIDlet es el conjunto de clases diseñadas para ser ejecutadas y controladas por la aplicación vía la interfaz. Se pueden tener varios MIDlets a la vez, y seleccionar cual está activo en un momento determinado llamando al método startApp() y ponerlo en pausa con la llamada a otro método, el pauseApp().

En el estado pasivo el MIDlet está inicializado y en reposo. En este estado no debería utilizar ningún recurso compartido. Al construir un MIDlet con el método new se realiza una llamada al constructor sin parámetros del MIDlet. En teoría no debe haber ningún problema, ya que pocas operaciones se suelen realizar en este estado. Si por alguna razón se lanza una excepción el MIDlet entraría inmediatamente en el estado destroyed y sería descartado.

Durante la ejecución normal del un MIDlet, éste se encuentra en el estado activo. Para pasar a este estado ha sido necesario realizar una llamada del tipo MIDlet.startApp().

Por último un MIDlet se encuentra en el estado destruido cuando su ejecución ha finalizado, se han liberado todos sus recursos. Cabe destacar que a este estado sólo se puede entrar una vez.

## *I.X Referencias*

**[J2ME]** http://java.sun.com/j2me/

# Apendice II Map Object-Java

En la implementación fue utilizado Map Object-java [MapObject], lo cual permitió representar mapas georeferenciados en la aplicación.

Map object-java es una suite de componentes desarrollados en java para crear aplicaciones GIS que contiene una colección de librerías (archivos jar). Estas librerías proveen gran cantidad de recursos GIS y componentes para visualización de mapas.

Por ser una herramienta de GIS, Map Object, provee funcionalidades y objetos que representan el dominio de aplicaciones GIS, que fueron descriptas en el capítulo 3.

Por este motivo, en el capítulo se mencionan abstracciones que se supone el lector conoce.

Los componentes que provee Map Object pueden ser divididos en las siguientes categorías:

- Objetos de visualización.
- Objetos de contenido.
- Objetos de acceso a datos.

## *II.I Objetos de visualización*

Esta sección describe los componentes de visualización utilizados para la implementación del prototipo de AVL.

### II.I.i Mapa

El mapa (Map) provee una forma de visualizar los datos espaciales. Cada tipo de datos es visualizado en un layer separado. Usualmente los layers mostrados se obtienen desde un origen (por ejemplo desde un path de disco), otros layers a visualizar pueden ser creados por el usuario y su repintado es dinámico, luego de su creación pueden ser guardados en disco.

La Figura II.1 muestra la componente de visualización map, con un layer cargado de disco desde un archivo con formato shapeFile de Esri (representa el mapa de la ciudad de La Plata) y con layers creados por el usuario (puntos que pueden visualizarse en el mapa).

**Figura II.1: Visualización de un mapa en la componente Map**

## II.I.ii Tabla de contenidos

La tabla de contenidos (TreeToc), provee una forma de visualizar los layer existentes en un mapa. Cada layer es representado por un nodo con una leyenda que contiene un color con el que se muestra el layer, un título y un checkbox, que indica la visibilidad. Para que el TreeTock reconozca sobre que mapa debe mostrar los layers debe enviársele el mensaje setMap(aMap).

La figura II.2 muestra una tabla de contenido, en donde pueden visualizarse dos layers, uno de manzanas (seleccionado para visualización) y otro de calles.

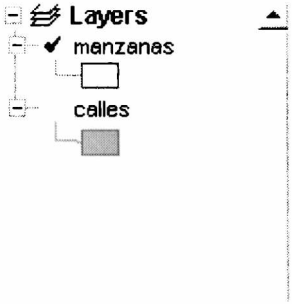


**Figura II.2: Componente TreeToc**

### II.I.iii Layers dinámicos

Un layer dinámico (AcetateLayer) es un componente transparente que es usado para crear layers que pueden encontrarse en continuo movimiento. Un ejemplo de uso es representar autos en movimiento. En un AcetateLayer pueden agregarse, removerse, o cambiarse elementos dinámicamente.

La diferencia básica entre un layer estático y un Acetatelayer, es que este último no necesita un repintado para mostrar cualquier actualización.

Cabe destacar que un mapa puede contener todos los AcetateLayer que sean necesarios.

### II.I.iv Herramientas

Las herramientas (Tools) permiten manejar eventos del mouse sobre el mapa. Solo una herramienta puede estar seleccionada a la vez. Un ejemplo de herramienta es la de zoom, que permite ver el mapa con mayor detalle.

La figura II.3 muestra como actúa la herramienta de zoom.

**Figura II.3: Ejemplo de funcionamiento de herramienta de zoom**

Como puede apreciarse la imagen que se encuentra a la derecha figura representa el resultado de hacer un "zoom in" sobre el mapa.

### II.I.v Herramientas de selección

Las herramientas de selección de elementos (RubberBands y RubberBandTool) permiten una selección de elementos. La forma en que la RubberBand es utilizada para realizar la selección depende de la subclase de RubberBand que sea utilizada.

Por ejemplo, una RubberBandEnvelope construye un rectángulo con la esquina inicial localizada en el mouseDown y la esquina final en el MouseUp.

La Figura II.4 muestra la selección a través de una RubberBandEnvelope

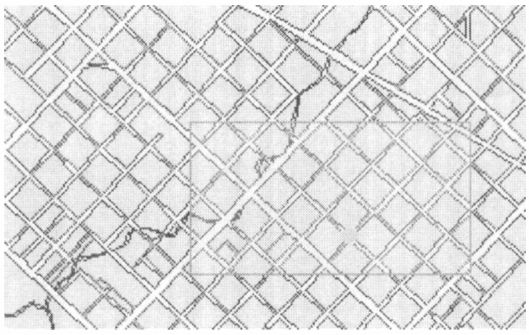

**Figura II.4: Selección a través de una herramienta.**

### II.I.vi Barras de herramientas

Una barra de herramientas (Toolbar) encapsula funcionalidades que definen acciones sobre un mapa. Una Toolbar debe conocer el mapa sobre el que va a actuar. Las toolbars son adaptables, ósea se le pueden agregar o quitar Tools para adaptarla a las necesidades.

MapObject-Java tiene preestablecido un conjunto de Toolbars que pueden ser reutilizadas.

La figura II.5 muestra la barra de herramientas ZoomPanToolBar adaptada con otras Tools.

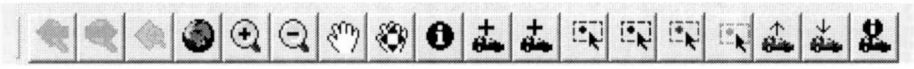

**Figura II.5: Barra de Herramientas adaptada.**

## *II.II Objetos de contenido*

Dada la extensa cantidad de objetos de contenido disponibles en MapObject-Java, en esta sección solo serán mencionados aquellos utilizados para la realización del prototipo de AVL.

### II.II.i Mapa

Así como existe el objeto Map para visualización, existe el objeto Map de contenido que simboliza un mapa. Es decir el mapa de visualización es una vista del mapa de contenido.

Para definir la abstracción de una mapa se definen un conjunto de interfaces para promover el rehusó de clases existentes, tanto para aplicaciones Client-Side como Server-Side. Las interfaces trabajan como un puente para compartir un protocolo común independientemente de cómo sea realizada la operación.

### II.II.ii Display

Un objeto Map de visualización tiene asociado un objeto DisplayManager que organiza la forma en que los layers son visualizados en el mapa. Un Display básicamente muestra tres tipos de objetos: figuras (shapes), selecciones y nombres. Un DisplayManager también es el encargado de crear el área que debe mostrarse, esto se

hace con la clase DisplayArea. Un DisplayArea representa un área particular que debe visualizarse del mapa con el que se esta trabajando.

### II.II.iii Layers

Un mapa esta compuesto por un conjunto de objetos layer que se organizan y manipulan usando un LayerSet. LayerSet es quien realmente contiene y ordena layers y cualquier clase de layer puede ser agregado a un LayerSet.

Así como existen layers simples existen GroupLayers que pueden ser usados para agrupar layers existentes en una estructura de árbol.

### II.II.iv FeatureLayer

Un FeatureLayer necesariamente debe mostrar las características y/o los nombres que se encuentran asociados a una FeatureClass, quien realmente contiene los elementos del layer. Existe un renderer separado (explicado posteriormente) para dibujar shapes (figuras que se muestran en el layer), selecciones (figuras seleccionadas) y los labels (textos asociados al layer). Cuando se debe dibujar un elemento el FeatureLayer colabora con el DisplayManager para invocar el renderer apropiado.

### II.II.v Renderers

Los renderers son los que tienen la responsabilidad de saber como dibujar un objeto contenido en un layer. Aunque los renderers son principalmente usados por FeatureLayer para mostrar componentes, estos también pueden ser usados por otros tipos de objetos como por ejemplo RubberBands.

Los renderers estándares son utilizados por un FeatureLayer para dibujar todas sus componentes de manera adecuada.

Existen varias subclases de renderers. Por ejemplo, GroupRenderer maneja varios Renderers y tienen una propiedad que permite escoger cual renderer utilizar en cada componente del layer.,ScaleDependentRenderer provee dependencias de escalas para un elemento, SimpleRenderer provee una forma de dibujar todas las componentes usando un solo símbolo.

También están implementados los LabelRenderer, que manejan símbolos para los textos del mapa y permiten dibujar uno o más campos. Si más de un campo es especificado, entonces el caracter blanco es usado para concatenar los valores.

### II.II.vi Símbolos

Un símbolo (Symbol) dibuja la geometría de características en un objeto gráfico 2D que es asociado con un visualizador de mapa. Cada Renderer tiene asociado al menos un Symbol que es el que va a utilizar para dibujar la componente de layer.

Varias implementaciones de Symbol se encuentran disponibles, donde cada una produce diferentes efectos visuales. Por ejemplo, un SimpleFillSymbol puede ser usado para rellenar polígonos con colores sólidos, y un RasterFillSymbol puede ser usado para dibujar polígonos rellenos con un patrón obtenido desde una imagen.

TextSymbols operan de forma similar a un símbolo para dibujar componentes de layer, pero estos solo sirven para dibujar labels dentro de un mapa.

GroupSymbols pueden ser usados para crear efectos de repintado compuesto para un componente.

### II.II.vii Layers gráficos

Un layer gráfico (GraphicLayer) es un contenedor que puede ser usado para mostrar diferentes clases de geometrías y nombres en un mismo layer. Este contrasta con el FeatureLayer porque el FeatureLayer solo puede mostrar un tipo de geometría (punto línea o polígono). Cualquier elemento del tipo Element (intereface) puede ser agregado al GraphicLayer.

Un GraphicLayer mantiene una lista de elementos que pueden ser seleccionados o buscados. Este provee métodos para cambiar el orden en que se muestran los elementos, tal como enviarlos al frente o enviar al fondo, y moverlos o removerlos.

Un FearureGraphicLayer provee soporte adicional para mostrar varios elementos a la vez, así como también localizar elementos usando criterios espaciales.

### II.II.viii Geometry

FeatureGeometry es la geometría asociada con componentes. Las tres familias de tipos asociadas a FeatureGeometry son Polígono, polilinea y multi puntos. Por razones de performance, el tipo punto no fue incluido en Map object.

Las FeatureGeometry están compuestas por otros elementos: un multipunto (MultiPoint) que contiene una colección de puntos (PointCollection), una polilinea (Polilyne) que contiene cero o más líneas (Paths), y un polígono (Polygon) que contiene elementos que lo forman (Rings). A su vez, Rings y Paths contienen una colección de puntos (PointCollection).

Para facilitar la forma de repintado las FeatureGeometry extienden java.awt.Shape.

### II.II.ix Objetos de acceso a datos

Esta sección introduce componentes que pueden ser usados para acceso a datos.

### II.II.x Content API

El Content API define una interfaz para acceso a datos GIS, desde diversos orígenes. Provee una conexión que brinda una forma transparente de acceder a orígenes de datos. Esta información es utilizada para crear un Source, quien maneja una conexión a recursos.

### II.II.xi Conjunto de datos de layer

Un conjunto de datos de layer (DataSetLayer) es el que realmente contiene todos lo elementos de un layer (Element).

Desde un LayerSource se pueden obtener DataSetLayer. Dependiendo el tipo de LayerSource varían los DataSetLayer. De esta forma para un FeatureLayer su Dataset es un FeatureClass y para un ImageLayer su Dataset es ImageClass.

La interfaz de LayerSource permite acceder de forma transparente a los elementos del layer sin tener que llegar al Dataset, solo se debe conocer la existencia del Dataset si se desea agregarle algún comportamiento extra al mismo.

### II.II.xii Consultas

Cuando se hace una operación de búsqueda sobre un layer, se debe indicar como se va realizar esa búsqueda, es para esto que existen los consultas (Queries).

Para seleccionar o buscar, se debe crear un filtro del Query (QueryFilter). Un QueryFilter debe contener una lista de campos que describen los valores que se quieren obtener desde la búsqueda, y opcionalmente puede tener una cláusula para selección de elementos (la cláusula de selección funciona como lo hace el where de SQL). Si además se quiere restringir que la búsqueda se haga en solo una porción del layer se debe crear un filtro espacial (SpatialFilter).

### II.II.xiii Datos

Cuando se hace una búsqueda sobre algún layer el resultado es un Cursor que contiene datos. Dependiendo el tipo del layer será el tipo de datos obtenido. Por ejemplo, un Featurelayer retorna componentes y un ImageLayer retorna imágenes.

### II.II.xiv Conexión con archivos

Es una implementación para una conexión con el file system. A partir de esta, una aplicación puede fácilmente usar un objeto conexión para conectarse con uno o varios archivos. Para la implementación de la conexión con el file system se provee un FileHandler que brinda una forma genérica de obtener el contenido de una FileFolderConection. Además, existe un jerarquía de handlers dependiendo del tipo de archivos con el que se desea trabajar, por ejemplo, para un archivo de formato shapefile de ESRI (.shp) se utiliza un ShapefileHandler.

### II.II.xv Índices Espaciales

Los índice espaciales (SpatialIndex) facilitan las búsqueda de datos espaciales en memoria, de forma de hacerla más eficiente.

Para indexación provee MutableTree y Tree, donde los nodos que pueden ser agregados en el árbol deben ser subclases de SpatialNode y si se busca recorrer un SpatialTree se debe crear un NodeVisitor.

### II.II.xvi Cache

La cache (MemoryCache) provee una forma de almacenar datos identificables en memoria para aumentar la velocidad de acceso a ellos. La cache tiene un número fijo de elementos identificables que pueden ser almacenados.

## *II.III Referencias*

**[MapObject]** http://www.esri.com/software/mojava/index.html.