TESINA DE GRADO

Universidad Nacional de La Plata

TITULO: "Desarrollo de Funcionalidad de un Browser para Hipermedia Física".

AUTOR: Racca, Juan Matías - Nº 0524/7.

DIRECTOR: Dra. Silvia Gordillo.

CARRERA: Licenciatura en Sistemas.

Resumen

En primera instancia se proveerá la funcionalidad de un browser que permita visualizar aplicaciones de Hipermedia Física. Para llevar a cabo esto se desarrollará una barra de navegación adecuada a las características de este tipo de aplicaciones.
La segunda contribución, es diseñar y desarrollar nuevas políticas de navegación que se ajusten a este tipo de aplicaciones, estas políticas modificarán el comportamiento de los botones back y next de un navegador. El desarrollo de nuevas políticas estará focalizado en la navegación física, mientras que se proveerán distintas políticas de navegación digital ya existentes.

Líneas de Investigación

Investigacion aplicada.

Trabajos realizados

Estudio y desarrollo de políticas de navegación digital y física.

Conslusiones

Se desarrolló la funcionalidad de navegación de un browser para aplicaciones de Hipermedia Física; esto dió como resultado el modelo de navegación para un browser de Hipermedia Física, logrando una clara separación entre el modelo de navegación y el browser. Junto al modelo de navegación se diseñaron nuevas políticas perzonalizadas de navegación, tanto digital como física.

Trabajos futuros

Incorporar mapas y caminos reales. Otro trabajo futuro podría ser extender los dominios presentados incorporando nuevos dominios navegacionales. También se podría pensar en extender la funcionalidad del browser, como podría ser permitir tener bookmarks.

Fecha de presentación: Agosoto 2008

Agradecimientos

Esta tesina no es el resultado de un simple trabajo individual, sino que es el resultado de una sumatoria de apoyos y esfuerzos a lo largo de muchos años. A todas las personas que, aunque no sean conscientes de ello, han hecho posible finalizar esta tesina, mil gracias de sincero corazón. Quiero agradecer en especial, dentro del ambiente académico, a Cecilia Challiol, sus "speedbacks" fueron increibles, muchas gracias.

A mis padres, Cristina & Félix, por darme sus genes, su sangre, su coraje... Ellos me brindaron la posibilidad de formarme académicamente, algo invaluable en estos tiempos que corren. Su apoyo y paciencia Incondicional fue fundamental, hoy dieron sus frutos. A mis hermanos, Carolina & Gabriel, alentaron a la distancia (desde Junín) y soportaron también. A mi locura divina, mi Nonna Lucía, todavía conservo aquella lechuza que me regalaste y que supo decirme, a diario, "al estudiar: perseverancia, paciencia y esfuerzo". A mi perro, Guten, como lo extraño. Solo resta nombrar, dentro de mi linaje, al mejor de todos (por varios cuerpos), mi Dios Antonio. Tu fuerza y pasión resultaron ser mi locomoción diaria, gracias por enseñarme a ser yo.

No puedo olvidar agradecer a todos mis compañeros de estudio (muchos mutaron en amigos) y amigos, para ellos mis saludos.

Por último y en especial, para la mejor copiloto de todas, Romi ("...She's a well to be dug, she's a university, a cosmic library..."). Ahora si, terminé la Tesina!

Índice General

Índice de figuras

Capítulo 1. Introducción.

1.1. Introducción a la tesis.

En los últimos años, el acceso a los dispositivos móviles ha ido creciendo de manera vertiginosa y, por el momento, no hay ninguna tendencia que indique que la misma no seguirá en alza. Los mismos están en continua evolución, a punto tal que hoy, los celulares, se han transformado en dispositivos multimedia de alta capacidad que procesan interfaces intuitivas de usuario. Por otro lado, cada vez son más las ciudades que poseen WIFI y acceso a Internet con dispositivos móviles, como algunas ciudades de Japón, Hong Kong y Corea; y también la futura Ciudad Ubicua (U-City, Ubiquitous City) en Corea del Sur, New Songdo City [O'Connell, 2005].

Una de las funcionalidades que está en desarrollo para las unidades móviles, es la de permitir al usuario del dispositivo, interactuar no sólo con el dispositivo a través de la navegación digital, sino también incorporar el espacio físico que lo rodea. La incorporación del espacio físico en las aplicaciones móviles, implica tener en cuenta la ubicación de la persona, para poder brindarle navegación e información acorde a lo que la persona está viendo. Es evidente que es mucho más atractivo aumentar el mundo digital incluyendo los objetos físicos que seguir manejando la información digital como un mundo virtual totalmente separado de la realidad. Esto motivo la integración de información física e Hipermedia Tradicional, dando como resultado la Hipermedia Física [Eriksen et al, 2003].

El término Hipermedia Física ha sido utilizado para describir cualquier sistema de hipermedia que trata con las propiedades del mundo físico. Podemos aplicar el concepto de Hipermedia Física en cualquier contexto (Museo, Exposición, Ciudad, etc), en donde los usuarios (sensados a través de un dispositivo móvil) pueden moverse físicamente en él. Dentro de Hipermedia Física encontramos dos tipos de links, los digitales y los físicos [Goble et al, 2004].

Los links digitales son los que encontramos dentro de una página web, son links a otros objetos digitales y su acceso es instantáneo (desde el punto de vista del usuario). A medida que el usuario va navegando mediante la selección de estos links va generando un historial de navegación. El historial de navegación puede recorrerse mediante los botones de back y next del browser. El funcionamiento de estos botones es determinado por la política de navegación [Greenberg et al, 2000] asociada al browser.

A diferencia de los links digitales, los links físicos son aquellos que unen dos objetos físicos (objetos del mundo real). Esto implica que cuando un usuario selecciona un link físico (manifiesta su intención de obtener información del target), recibe información para llegar hasta el objeto que representa el target del link. Es decir, el usuario debe caminar hasta el target para visualizar su información, esto se conoce como “caminar un link” (“Walk the link”, [Goble et al, 2004]). Cuando el usuario “camina un link” lo que está haciendo es realizar una navegación física. Una navegación física implica que el usuario seleccione un link físico y camine siguiendo las instrucciones del sistema para alcanzar el objeto target, cuando el

usuario llega al objeto target recién en ese momento la navegación física se considera finaliza.

Es decir, que el usuario va navegando físicamente de un objeto físico a otro para obtener información. Pero que pasa si en un determinado momento el usuario quiere retroceder a un objeto físico que había visitado. El usuario tendría que tener la misma posibilidad de recorrer su historial de navegación física (como el browser lo permite en el caso de la navegación digital), pudiendo retroceder a objetos ya visitados.
Esto motiva el objetivo de esta tesis que es brindar políticas de navegación física para Aplicaciones de Hipermedia Física. Y Además brindar un browser que tenga definido los botones de back y next tradicionales y además su contraparte física (botones de back y next para la navegación física).

1.2. Contribuciones.

Las contribuciones que se podrían desprender de esta tesis son:

- En primer lugar, desarrollar la funcionalidad de navegación de un browser para aplicaciones de Hipermedia Física.
- En segundo término, y como complemento de la parte principal, diseñar políticas personalizadas de navegación permitiendo incorporar conceptos específicos según cada dominio.

En primera instancia se proveerá la funcionalidad de un browser que permita visualizar aplicaciones de Hipermedia Física. Para llevar a cabo esto se desarrollará una barra de navegación adecuada a las características de este tipo de aplicaciones.

La segunda contribución, es diseñar y desarrollar nuevas políticas de navegación que se ajusten a este tipo de aplicaciones, estas políticas modificarán el comportamiento de los botones back y next de un navegador. El desarrollo de nuevas políticas estará focalizado en la navegación física, mientras que se proveerán distintas políticas de navegación digital ya existentes.

1.3. Estructura.

El resto de la tesis se organiza de la siguiente manera: en el Capítulo 2 se realizará una introducción a los temas de Hipermedia, Hipermedia Física, Browser de Hipermedia y Aplicaciones Sensibles al Contexto a fin de conocer el ambiente de la aplicación a desarrollar. En el Capítulo 3 se darán las bases para el desarrollo junto a una introducción de objetos digitales y físicos, como también un modelo de navegación abstracto. Una vez comprendido el ambiente y las bases para el desarrollo, en el Capítulo 4, se desarrollará la funcionalidad de un Browser para aplicaciones de Hipermedia Física dividiendo el desarrollo en dos grandes etapas, una digital y otra física. Una vez finalizado el desarrollo se presentarán, en el Capítulo 5, ejemplos de navegaciones digitales y físicas. El Capítulo 6 cierra esta tesis con conclusiones alcanzadas.

Capítulo 2. Hipermedia: Definiciones.

2.1. ¿Que es Hipermedia?

El término Hipermedia toma su nombre de la suma de hipertexto y multimedia. A continuación se analiza cada concepto, finalizando con la definición de Hipermedia.

2.1.1. Hipertexto.

El término Hipertexto [Nelson, 1965] es utilizado por primera vez por Theodor Nelson para designar un sistema que permitía la lectura y escritura no lineal. Lo define como:

> "un cuerpo de material escrito o pictórico interconectado en una forma compleja que no puede ser representado en forma conveniente haciendo uso de papel".

El Hipertexto es una tecnología de la información que imita la organización asociativa de la memoria humana. Este fragmenta la información en bloques de contenido (nodos) que se conectan a través de enlaces (links), cuya selección provoca la inmediata recuperación de la información destino. El mismo permite navegar por un entramado de nodos, de acuerdo con las preferencias o las necesidades que se tengan en cada momento.

El Hipertexto sirve para representar aplicaciones que tengan gran volumen de información y donde generalmente los usuarios sólo necesitan una parte de ella. Un requisito fundamental es que la información debe estar fragmentada y que los fragmentos tengan relaciones entre sí.

La estructura del Hipertexto es una estructura bastante compleja que integra en sí misma varios tipos distintos de organización de la información: secuencial, jerárquica y en red. En la Figura 1 podemos comprobar que el estilo hipertextual es un tipo de estructura que conjuga todos ellos, es decir, una estructura hipertextual incluye (o puede incluir) tanto estructuras secuenciales, como reticulares y jerárquicas.

Figura 1: Estilos de Hipertexto.

La forma más simple de Hipertexto es la unión entre nodos, también llamado Hipertexto de nodos encadenados (ver Figura 2). Este tipo de hipertexto funciona como un glosario de acceso aleatorio que posibilita el acceso directo a cualquier nodo en el hipertexto.

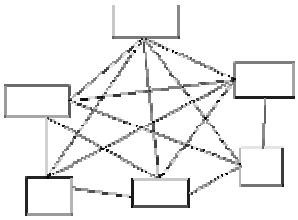


Figura 2: Hipertexto de nodos encadenados.

Algunas definiciones de Hipertexto:

- "El hipertexto es una tecnología que organiza una base de información en bloques distintos de contenidos, conectados a través de una serie de enlaces cuya activación o selección provoca la recuperación de información". [Díaz et al, 1996].
- "...el uso del computador que trasciende la linealidad, límites y calidad fija de la tradicional forma de escritura de texto". [Landow et al, 1991].
- "hipertexto consiste de tópicos y sus conexiones; los tópicos pueden ser párrafos, oraciones o palabras simples. Un hipertexto es como un libro impreso en el cual el autor tiene disponible un par de tijeras para cortar y pegar pedazos de redacción de tamaño conveniente. La diferencia es que el hipertexto electrónico no se disuelve en una desordenada carpeta de anotaciones: el autor define su estructura definiendo conexiones entre esas anotaciones". [Bolter, 1991].
  Existen otros autores que también detallan definiciones de Hipertexto como son [Conklin, 1987], [Fiderio, 1988], [Balzer et al, 1989], [Rada, 1991].

2.1.2. Multimedia.

Es difícil definir en pocas palabras el término multimedia. Se puede decir que, en una computadora, es la capacidad de mostrar gráfico, vídeo, sonido, texto y animaciones como forma de trabajo, e integrarlo todo en un mismo entorno llamativo para el usuario, que interactuará o no sobre él para obtener un resultado visible, audible o ambas cosas. En efecto, las riquezas de los multimedios reside en el acopio de información.

El concepto de Multimedia es amplio, a continuación se hace mención a algunos conceptos declarados a través de los años:

- "…Permite a los aprendices interactuar activamente con la información y luego reestructurarla en formas significativas personales. Ofrecen ambientes ricos en información, herramientas para investigar y sintetizar información y guías para su investigación…" [Schlumpf, 1990].

- "…Intento de combinar la capacidad autoexplicativa de los medios audiovisuales con el texto y fotografías para crear un medio nuevo de comunicación único en la pantalla de la computadora…" [Lynch, 1991].
- "…Integración de dos o más medios de comunicación que pueden ser controlados o manipulados por el usuario mediante la computadora; video, texto, gráficos, audio y animación controlada con una computadora; combinación de hardware, software y tecnologías de almacenamiento incorporadas para proveer un ambiente de información multisensorial…" [Galbreath, 1992].

2.1.3. Hipermedia.

El concepto de Hipermedia tiene su origen en la combinación de HIPERtexto y multiMEDIA. Por lo tanto, es la tecnología que nos permite estructurar la información de una manera no-secuencial, a través de nodos interconectados por enlaces. La información presentada en estos nodos podrá integrar diferentes medios (texto, sonido, gráficos, etc).

La Hipermedia conjuga los beneficios de ambas tecnologías: La Multimedia proporciona mayor riqueza en la transmisión de información y el Hipertexto aporta una estructura que permite que los documentos puedan ser explorados y presentados siguiendo diferentes secuencias. Por ejemplo: Enciclopedias, Museos virtuales, Periódicos en Internet, Películas en formato DVD.

A continuación, se enuncian algunas diferencias claves con Hipertexto:

- Aparición de informaciones de carácter dinámico (su presentación tienen una duración que no coincide con la del nodo, por ejemplo un sonido) y reactivo (permiten la interactuación).
- Necesidad de una gran cantidad de espacio para almacenar información relacionada con los distintos medios.

Ventajas de la Hipermedia respecto al Hipertexto:

- Tienen interfaces de usuario muy ergonómicas e intuitivas.
- El usuario no necesita realizar grandes esfuerzos para conseguir rápidamente resultados.
- La información se recupera sencillamente, aunque distintos usuarios estén utilizando el mismo documento simultáneamente.
- Permiten representar información poco o nada estructurada.
- La creación de nuevas referencias es inmediata, independientemente del tipo de contenido involucrado.
- Posibilitan la estructuración de la información.
- Facilitan la modularidad y la consistencia de la información.
- Constituyen un marco idóneo para la autoría en colaboración, al permitir la compartición, distribución y personalización de la información.
- Permiten acceder a la información secuencialmente, navegando o planteando consultas en un lenguaje de interrogación, según las necesidades de cada usuario.

2.2. ¿Que es Hipermedia Física?

2.2.1. Definición e historia.

El primer paradigma que apareció para unir el mundo virtual con el real fue la Realidad Aumentada (Augmented Reality) [Mackay, 1998]; en lugar de aumentar el mundo virtual con información real (Virtual Reality), la meta es aumentar los objetos reales por medio de objetos digitales y la capacidad de comunicación. La Realidad Aumentada mantiene el mundo real del usuario enriqueciéndolo con la presencia de elementos virtuales.

La Realidad Aumentada es una variación de la Realidad Virtual. En esta última, el usuario es sumergido totalmente en un mundo virtual, dentro de éste no puede ver el mundo real que lo rodea. En contraste, la Realidad Aumentada permite al usuario ver el mundo real con los objetos virtuales superpuestos sobre él. La gran diferencia es que complementa el mundo real, no lo sustituye por uno virtual; el usuario percibe que tanto objetos reales como virtuales coexisten en un mismo espacio.

La Realidad Virtual y la Realidad Aumentada están muy relacionadas, pero albergan diferencias:

- La realidad virtual permite la inmersión del usuario en un mundo artificial que sustituye completamente al mundo real del usuario.
- La realidad aumentada mantiene el mundo real del usuario enriqueciéndolo con la presencia de elementos virtuales.

Ambas realidades son un subconjunto de un tipo de aplicaciones conocidas como Mixed Reality (MR) [Milgram et al, 1994] (ver Figura 3).

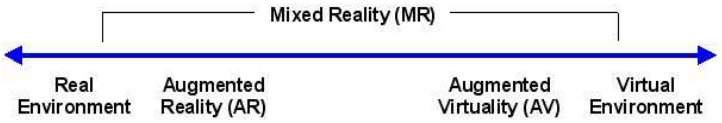


Figura 3: Explicación de P. Milgram sobre MR [Milgram et al, 1994].

Mixed Reality se refiere a la mezcla de mundos reales y virtuales para producir nuevos ambientes y visualizaciones en donde objetos físicos y digitales coexisten e interactúen en tiempo real. Una mezcla de realidad, realidad aumentada, virtualidad aumentada y realidad virtual.

Las posibles aplicaciones tienen cabida en numerosos campos, entre ellos Medicina, Militar, Educación, Juegos y entretenimientos.

Por ejemplo, con esta tecnología (Realidad Aumentada) es posible que un mecánico detecte un fallo en el motor sin necesidad de desarmarlo, ya que mediante la Realidad Aumentada puede apreciar la totalidad de su estructura con sólo resaltar en su video casco las secciones marcadas en el motor. En la Figura 4 se puede apreciar la vista que tendrá el mecánico al momento de reparar o

desarmar un vehículo, en este ejemplo en particular se muestra como desarmar ciertas piezas del mismo por medio de realidad aumentada.

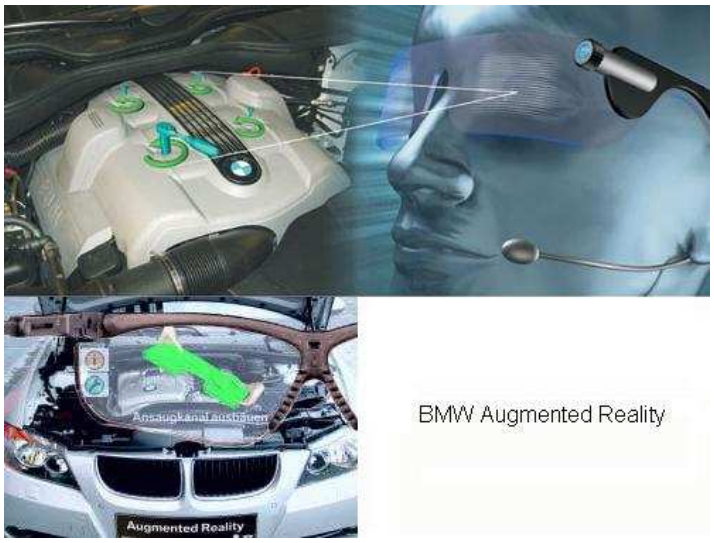


Figura 4: RA aplicada en la mecánica [BMW, 2008].

Lo mismo puede decirse de la medicina, que gracias a esta tecnología puede observar los tejidos del cuerpo mediante una ampliación virtual del organismo, sin necesidad de recurrir a los rayos X. Más recientemente, incluso se ha aplicado la tecnología de la Realidad Aumentada para el mantenimiento de una planta nuclear. La Realidad Aumentada también puede aplicar en otras áreas, como ser la reconstrucción de edificios antiguos o en ruinas.

Más recientemente, sin embargo, se explora la posibilidad de utilizar pequeños dispositivos móviles como interfaz de los sistemas de Realidad Aumentada. PDAs y teléfonos móviles con cámaras podrían convertirse también en eficaces instrumentos de Realidad Aumentada, tal como ocurre con el juego del Tren Invisible ("The Invisible Train", [Wagner et al, 2005]). En la Figura 5 se puede apreciar como juegan dos participantes.

Figura 5: RA aplicada en el juego "Tren Invisible" [Wagner et al, 2005].

Este juego es la primer aplicación de Realidad Aumentada multiusuario para PDAs. El Tren Invisible es un juego en el que los usuarios controlan trenes virtuales sobre una pista de ferrocarril miniatura real. Estos trenes virtuales solo son visibles a los usuarios a través de los videos de sus PDAs, no existen en el

mundo físico real. Este tipo de interfaz de usuario es comúnmente llamada “magic lens metaphor”. Los usuarios pueden interactuar en el ambiente del juego manejando los cambios de pista (agujas) y ajustando la velocidad de sus trenes virtuales. El estado actual del juego es sincronizado entre todos los participantes vía red inalámbrica. El objetivo principal del juego es evitar que los trenes virtuales colisionen entre si.

Un subconjunto de aplicaciones particulares de Realidad Aumentada, son las aplicaciones de Hipermedia Física.

La Hipermedia Física (Physical Hypermedia) surge de la necesidad de agregar información física a la Hipermedia tradicional, el término de Hipermedia Física ha sido acuñado en [Grønbæk et al, 2003], cuya meta es extender el paradigma de Hipermedia con la manipulación de objetos del mundo real. Hipermedia Física modela el espacio físico utilizando el paradigma de Hipertexto, permitiendo navegar al mismo tiempo por objetos físicos y digitales integrados en un mismo sistema. El espectro o dominio en donde se puede aplicar la Hipermedia Física es muy amplio; en muchas actividades las personas tratan con información tanto digital como física, como ser museos, fábricas, depósitos, hospitales.

A modo de ejemplo, podemos citar el sistema Topos, desarrollado en el proyecto WorkSPACE (ver Figura 6). El objetivo principal de este sistema es aumentar el ambiente de trabajo a través de Hipermedia Física. El concepto central en Topos es el de *espacios de trabajo* que es el medio primario para agrupar y organizar materiales y documentos en un espacio 3D. Los espacios de trabajo mismos pueden ser agrupados, mezclados y conectados de varias maneras.

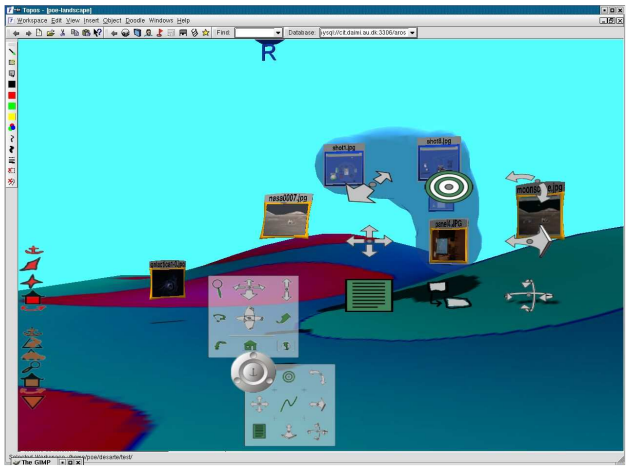


Figura 6: Cliente Topos para Linux (workSPACE) [Grønbæk et al, 2003].

Otro proyecto interesante sobre Hipermedia Física es el framework HyCon [Bouin et al, 2003] (y amplían su funcionamiento en [Hansen et al, 2004]) para sistemas de Hipermedia context-aware. Su arquitectura da soporte para realizar anotaciones, manejo de links y tours guiados asociando ubicaciones y objetos con dispositivos RFID o Bluetooth con mapas, páginas web y colecciones de recursos. Introduce el uso de XLinks y un nuevo sistema de búsqueda para la web llamado Geo-Based Search (GBS). Lo interesante de este sistema es que soporta los

mecanismos clásicos de Hipermedia para navegar, buscar, hacer notaciones y tours guiados en el mundo físico, permitiendo a los usuarios realmente unir objetos digitales y/o físicos. Cabe destacar también que esta arquitectura propuesta, incluye interfaces para incluir la capa de sensores, que encapsulan el GPS y otros sistemas de posicionamiento que se pueden utilizar. En la Figura 7 se ve un ejemplo de búsqueda y navegación en el prototipo HyConExplorer. Los puntos de color que se ven en el mapa digital son marcadores de enlace ("link markers") para información: ubicaciones anotadas, documentos linkeados y resultados de búsquedas. Activando un marcador de enlace se presenta la información vinculada en el cuadro a la derecha de la ventana.

Figura 7: Ejemplo de búsqueda y navegación en HyConExplorer [Bouin et al, 2003].

2.2.2. Característica de Hipermedia Física.

Como se enunció anteriormente, la Hipermedia Física relaciona objetos físicos con el espacio hipermedial. En la Figura 8 se muestra una representación gráfica de Hipermedia Física y su relación con la Web.

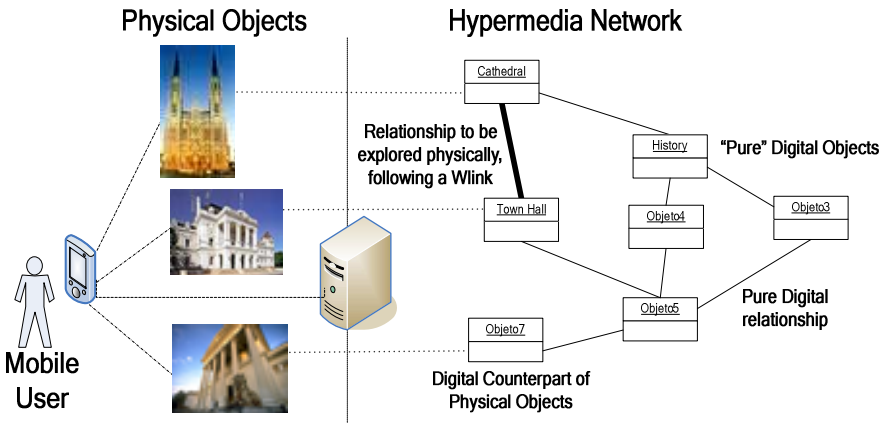


Figura 8: Hipermedia Física y la Web [Rossi et al, 2006].

A continuación se describe un sistema de Hipermedia Física basándose en el Método de Diseño de Hipermedia Orientado a Objetos ("Object Oriented Hypermedia Design Methodology", OOHDM) [Challiol et al, 2006]. Se define una aplicación de Hipermedia Física como una aplicación de hipermedia en donde todos o algunos de los objetos de interés son objetos del mundo real que son visitados físicamente por usuarios. El escenario más usual para estas aplicaciones involucra un usuario móvil y algún mecanismo de sensado de ubicación con el software subyacente que pueda determinar, por ejemplo, cuando un usuario está dentro del área de sensado de uno de esos objetos. En la Figura 9 vemos el enfoque OOHDM junto con los hot-spots en el cuál se modelan las características específicas de Hipermedia Física.

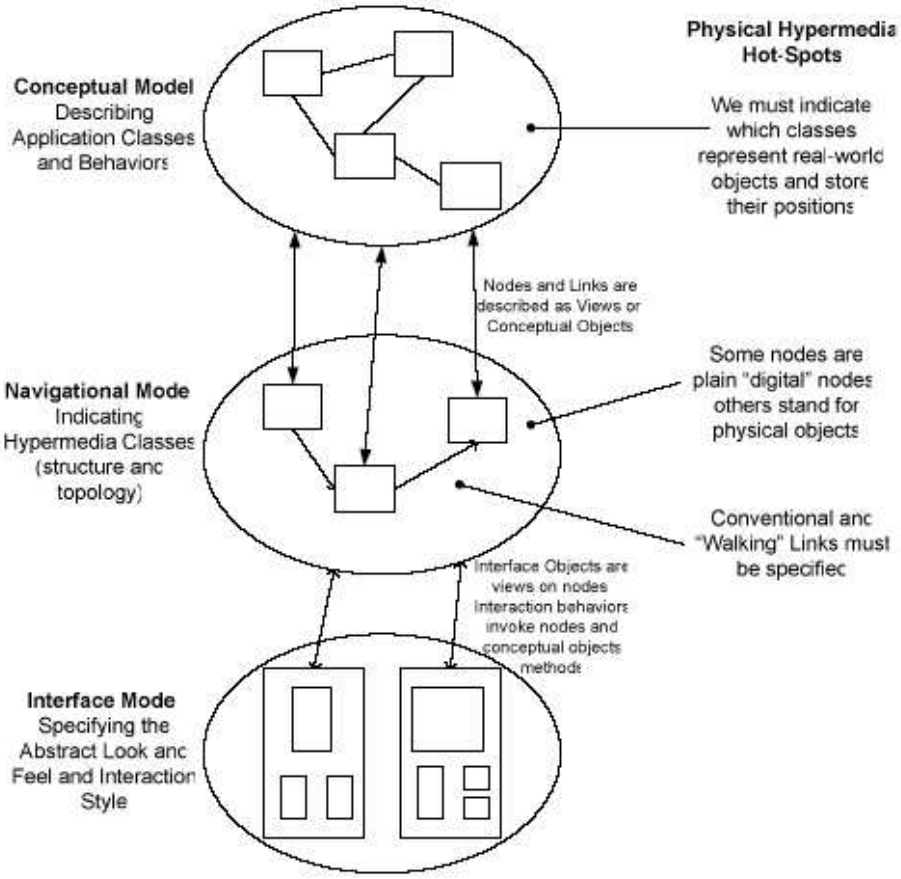


Figura 9: El enfoque OOHDM para Hipermedia Física [Challiol et al, 2006].

Dentro del modelo Conceptual existen dos tipos de Objetos, el Digital y el Físico. Un Objeto Digital es un componente Web tradicional y uno Físico es un objeto del mundo real, como ser un cuadro, una escultura, un edifico, etc.

Siguiendo con la línea de la Figura 9, nos encontramos con el modelo Navegacional, dentro de éste se especifican los Nodos que el usuario podrá explorar junto con los links que los conectan. Estos Nodos podrán ser tanto Digitales como Físicos.

Un Nodo Digital lo podemos subclasificar en Contraparte Digital (de un Nodo Físico) o Nodo Digital Puro. La Contraparte Digital son elementos digitales que

aumentan al Nodo Físico. A diferencia de una Contraparte Digital, los Nodos Digitales Puros no se asocian a ningún Nodo Físico.

Los links entre Nodos Digitales, llamados links digitales, permiten navegaciones convencionales (por ej: como en la Web), lo que significa que la semántica de estos links es la misma que la de un link convencional de Hipermedia (por ej: cuando un link es activado, se abre el Nodo destino).

Un Nodo Físico representa un Objeto Físico junto con sus interrelaciones (Links Físicos) a otros Objetos Físicos dentro de un Sistema de Hipermedia Física. Los Links Físicos o Caminables (WLinks – "Walking Links") expresan una relación en donde el objeto apuntado (destino) es un Objeto Físico, por lo tanto, explorar este objeto implica que el usuario deberá cambiar su posición actual. Cuando el usuario activa un WLink está indicando su intención de explorar ese destino; la respuesta del sistema podría ser un mapa o un plano que lo guíe al mismo. Luego él debe caminar a través del link y cuando llega a su destino, será sensado y el correspondiente Nodo se abrirá.

Los Links Físicos deberían derivar de las relaciones conceptuales y geográficas; por ejemplo, mientras estamos explorando un monumento, podríamos tener WLinks a otros objetos de interés "cercanos" (que significa una relación geográfica) o a otros objetos que tengan una relación semántica más fuerte, como ser un monumento construido por el mismo arquitecto.
Tomando estos conceptos, podemos encontrar, dentro de un sistema de Hipermedia Física, distintos tipos de relaciones entre nodos:

- Digital-Digital: es una relación equivalente a un link de hipermedia en la web.
- Física-Física: es una relación mediante alguna semántica de dos objetos físicos, esta relación permite al usuario poder llegar caminando de un objeto a otrro.

Es decir, que podríamos ver estas relaciones como red (grafo) de nodos interconectados por un lado la red de nodos digitales y por otro lado la red de nodos físicos. Estos dos tipos de relaciones permiten la navegación digital y física por parte del usuario. En el caso de que la conexión sea digital se utiliza el conocido paradigma de navegación de la Web (www) para desplazarse en el sistema, mientras que, cuando la relación es física, la conexión tiene que ser "caminada" por el usuario [Goble et al, 2004].

2.2.3. Resumen de Hipermedia física.

El paradigma de Hipermedia Física surge de la necesidad de agregar información física a la Hipermedia tradicional. Permite construir relaciones significativas utilizando el paradigma de Hipermedia tradicional. Podemos entender a la Hipermedia Física como un conjunto de Nodos Digitales y Físicos interconectados en donde cada Nodo Físico puede ser enriquecido con información digital (relación a un Nodo Digital). Cuando un usuario es sensado por un objeto del mundo real, se le mostrará en su dispositivo móvil información, tanto física como digital, así como también links físicos y digitales. Si el usuario selecciona un link físico, este

deberá caminar a través del link (Por ejemplo: mediante la información de un mapa) hasta llegar a su destino.

2.3. Browser de Hipermedia

2.3.1. Funcionalidad de un Browser de Hipermedia.

Un navegador web (browser) es una herramienta que permite al usuario recuperar y visualizar documentos de hipertexto e hipermedia, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet. Esta red de documentos es denominada World Wide Web (WWW). Cualquier navegador actual permite mostrar o ejecutar gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto e hipervínculos o enlaces.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Los documentos pueden estar ubicados en la computadora en donde está el usuario, pero también pueden estar en cualquier otro dispositivo que esté conectado a la computadora del usuario o a través de Internet, y que tenga los recursos necesarios para la transmisión de los documentos (un software servidor web). Tales documentos, comúnmente denominados páginas web, poseen hipervínculos que enlazan una porción de texto o una imagen a otro documento, normalmente relacionado con el texto o la imagen.

El seguimiento de enlaces (links) de una página a otra, ubicada en cualquier computadora conectada a la Internet, se llama navegación (cuando hablamos de navegación en un browser de Hipermedia tradicional nos referimos a navegación digital); que es de donde se origina el nombre de navegador. Un navegador o browser posee botones standard; entre los mas comunes se encuentran los botones de Back, Next, Reload y Stop. A continuación se describe brevemente la semántica de cada uno:

- Back: regresa a la página anterior visitada, normalmente la más reciente, aunque se puede regresar directamente a una página específica.
- Next: avanza a la página siguiente visitada (según el historial de navegación). Si el usuario no ha utilizado Back en su sesión actual, el botón Next permanece inactivo.
- Reload: vuelve a cargar la página actual. Esta función se utiliza si la información de la página actual cambió o se cargó incompleta, mostrando la versión más reciente de la página.
- Stop: interrumpe el acceso o carga de la página actual. Es común utilizarlo cuando se demora demasiado en cargar una página web.

2.3.2. Políticas de Navegación en un browser Hipermedia.

Los usuarios frecuentemente utilizan el botón de Back de un navegador Web para retornar a páginas recientemente visitadas. Dicho botón abarca más del 40% de las acciones de los usuarios en los navegadores web.

El modelo navegacional debajo de los botones Back y Next mas utilizado por los navegadores comerciales (firefox, iexplorer, opera) está basado en una pila de páginas web visitadas (stack-based). Esta política de navegación posee tres diferentes tipos de operaciones [Greenberg et al, 1999]:

1. Al clickear o tipear links se agrega una página al tope de la pila.
2. Clickeando los botones Back y Next se mueve el puntero (de la pila) hacia abajo y arriba respectivamente dentro de la pila, mostrando la página apuntada. El contenido actual de la pila no se altera al navegar con estos botones.
3. Cuando el usuario esta en cualquier posición de la pila, a excepción del tope, y selecciona un link en una página web, todas las entradas en la pila que están por encima de la posición actual se desapilan de la misma antes que se agregue la nueva página. Las páginas que fueron desapiladas no podrán ser revisitadas por medio de los botones de Back y Next.

El comportamiento stack-based no asegura que todos los nodos o páginas visitadas recientemente sean accesibles por medio del botón de Back, prestando a confusión por parte del usuario que llega a preguntarse porque se pierden páginas visitadas anteriormente (comportamiento mencionado en el punto 3).

A continuación se mostrará un ejemplo del modelo navegacional stack-based [Greenberg et al, 1999]: Supongamos un usuario que sigue los links desde la página A hasta D en orden (según la estructura especificada en la Figura 10a), luego oprime dos veces el botón de Back hasta retornar a B y luego selecciona un nuevo link, desde B, a H. La Figura 10b muestra como va quedando la pila, como se puede apreciar, las páginas C y D no son más accesibles por medio de los botones de Back y Next.

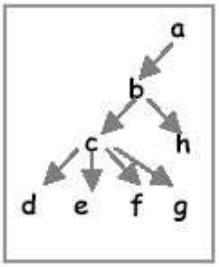


Figura 10a: Estructura de página.

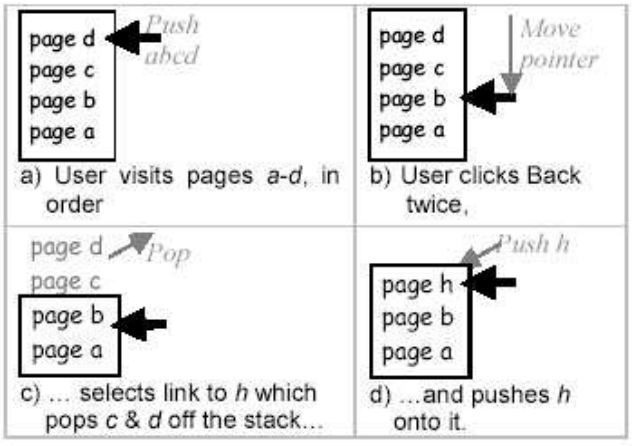


Figura 10b: Ejemplo de una navegación y su efecto en la pila. Notar que las páginas C y D son desapiladas de la misma.

Resumiendo, las ventajas de la política stack-based son:

- Implementa un mecanismo simple y requiere solo dos botones.

- El "podado" automático de páginas previamente visitadas podría eliminar páginas que el usuario ya no necesite
- Los usuarios son capaces de utilizar el mecanismo aún sin entenderlo.

Las desventajas de la política stack-based son:

- Los usuarios no pueden retornar a páginas que fueron desapiladas.
- El modelo no es intuitivo para el usuario. Consecuentemente, los usuarios se sorprenden cuando las páginas "desaparecen".
- Introduce dos semánticas diferentes para el mostrado de páginas. Selección de links; y revisitado de páginas por medio de los botones de Back y Next.

Quizás la mayor desventaja de la política de navegación stack-based es que no provee un historial completo de las páginas visitadas previamente. Cabe aclarar que, cuando se hace mención al historial de páginas visitadas no se hace referencia al historial navegación real, sino a la lista o pila que es utilizada por las distintas políticas de navegación; lo cuál no implica que si una página no es accesible por medio de los botones Back y Next, no sea accesible por el historial de navegación real. El historial de navegación real guarda todas las páginas que fueron visitadas por un usuario pero es independiente del modelo navegacional.

Como alternativa a la política stack-based, [Greenberg et al, 1999] provee un historial completo de las páginas visitadas previamente teniendo una lista de historial con base-reciente (recency-based), donde los botones de Back y Next naveguen a través de las páginas en orden reverso a como fueron vistas. Sorprendentemente, el diseño de un mecanismo para tener una lista de historial completa no es tan sencillo. Aunque armar una lista de todas las páginas que un usuario visitó resulte trivial, diseñar un mecanismo simple y comprensible para atravesar dicha lista es complejo. En [Greenberg et al, 1999] se enuncian y exploran varios modelos de Back basados en variantes de recency-based. A continuación se describirá solo el último de esos modelos, reciente con duplicados eliminados y mejora de ordenamiento temporal.

En primer lugar, veremos el manejo de las entradas duplicadas. Permitir páginas duplicadas significa que el sistema puede ofrecer una representación literal del orden en el que las páginas fueron visitadas. La desventaja de esto es que la lista podría ser innecesariamente extensa y repetitiva. En vez de esto, se eliminan los duplicados manteniendo solo una copia en su posición mas reciente de la lista: esto mantiene las páginas revisitadas mas recientes cerca del tope. En [Tauscher et al, 1997] se analiza esto, y concluye que, solo se necesitaría algunos Back para retornar a una página deseada (sin duplicados).

Al igual que en stack-based, navegar a través de links agrega una página al tope de la lista, y seleccionando Back o Next mueve el puntero a través del historial (mostrando la página en cada click). La diferencia está al seleccionar un link a una nueva página después de realizar un Back; aquí es donde se utiliza un algoritmo de ordenamiento temporal. Al seleccionar un nuevo link, después de haber navegado por medio de Back y Next, se debe reordenar la lista al verdadero orden temporal que encaje con lo que el usuario estuvo viendo o visitando (excluyendo los duplicados). Esta técnica se implementa manteniendo una lista secundaria

reciente que marca el orden de páginas visitadas cuando una persona navega la lista de historial primaria utilizando Back y Next. Tan pronto como el usuario clickea en una nueva página, el contenido de la lista secundaria es agregado en orden a la lista primaria.

En la Figura 11 se ve el resultado de procesar el mismo ejemplo citado anteriormente (en la Figura 10a y 10b) pero con un modelo navegacional recency-based [Greenberg et al, 1999]: tal como se mencionó en el ejemplo anterior, el usuario visita las páginas A, B, C y D (ver Figura 11a); luego retrocede, por medio del botón de Back, de D a B generando una segunda lista (ver Figura 11b). Tan pronto como el usuario selecciona un nuevo link, de B a H, C y D se agregan a la lista principal y se eliminan duplicados (ver Figura 11c). Luego se agrega H al tope de la lista (ver Figura 11d).

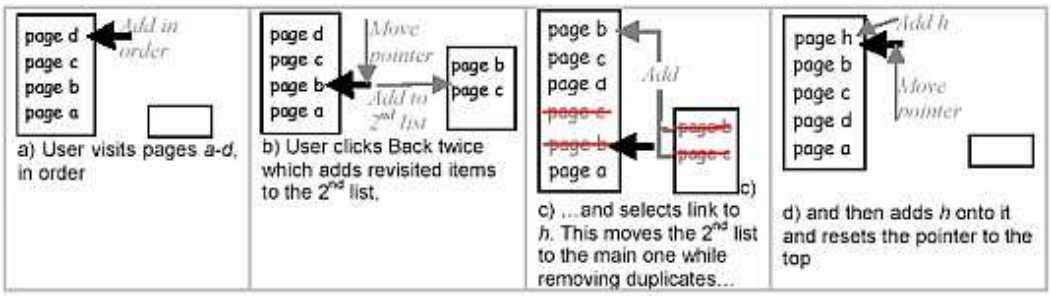


Figura 11: Ejemplo de recency-based [Greenberg et al, 1999].

Las ventajas de este modelo recency-based son:

- Conserva los botones standard de Back y Next.
- La lista de páginas visitadas previamente está completa porque ninguna página es desapilada, como sucede en stack-based. Por lo tanto, se garantiza a los usuarios poder acceder a cualquier página visitada anteriormente.
- Debido a que la lista de recientes subyacentes puede crecer indefinidamente, es factible para el Back poder trabajar mediante sesiones del browser.
- El algoritmo de reordenamiento temporal permite que los usuarios accedan de manera inversa a las páginas visitadas cuando usan el Back.

Las desventajas de este modelo son:

- Si la meta del usuario es navegar hacia atrás a través de la estructura del árbol que se fue generando producto de la navegación, el no realizar el podado de las ramas hace que dicha estructura se trate de manera lineal. Por lo tanto, el usuario no puede detectar cuando se abrió una nueva rama (producto de clickear un link nuevo cuando se realizo un back). Este método combina movimientos a través de una lista (por medio de Back y Next) con ordenamientos temporales (por medio de nuevas selecciones de links). Hay un riesgo de que los usuarios encuentren en el reordenamiento de páginas una lista impredecible o inesperada. Esto no debería ser un gran problema ya que la mayoría de los usuarios utilizan los botones de

Back y Next de modo mecánico, clickeando repetidamente los botones hasta reconocer la página buscada.

Las dos políticas enunciadas en este capítulo (stack-based y recency-based) son políticas digitales. Es decir, son utilizadas por un browser de Hipermedia para poder acceder, por medio de los botones de Back y Next, a las páginas anteriormente visitadas. Estas políticas van a ser implementadas en el Borwser para Hipermedia Física desarrollado en esta tesis, más específicamente, en la parte de Navegación Digital.

2.4. Aplicaciones sensibles al contexto.

La definición de contexto [Dey et al, 1999] que se tendrá en cuenta será la siguiente:

> "Contexto es cualquier información que puede ser utilizada 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 el usuario y la aplicación misma".

Casi toda la información disponible al tiempo de una interacción puede ser vista como información de contexto. Algunos ejemplos, información espacial (ubicación, orientación, velocidad), información temporal (hora del día, día), información del ambiente (temperatura, calidad del aire, ruido), actividad (caminando, leyendo, corriendo, hablando).

Un sistema es sensible al contexto si puede extraer, interpretar y utilizar información de contexto y adaptar su funcionalidad a su contexto actual. El reto de estos sistemas radica en la complejidad de capturar, representar y procesar datos del contexto. Según [Schilit, 1994], un desafío de la computación móvil es explotar los cambios de ambientes con una nueva clase de aplicaciones que son sensibles al contexto en el cuál son ejecutadas. Tales aplicaciones sensibles al contexto (context-aware) se adaptan de acuerdo a su posición, a las personas y a los objetos cercanos, así como también a los cambios que le ocurren a estos objetos en el tiempo. Un sistema con estas capacidades puede examinar el ambiente computacional y reaccionar a los cambios del ambiente.

Existen algunas características que son comunes en aplicaciones sensibles al contexto [Schilit, 1994]:

- Información y servicios pueden ser presentadas al usuario de acuerdo a su contexto actual: un ejemplo de información podría ser donde está el banco más cercano, uno de servicios podría ser una interfaz de usuario que cambie de comandos de acuerdo a la hora del día o la ubicación.
- Ejecución automática de un servicio cuando se está en un cierto contexto: esto incluye acciones disparadas de contexto (context-triggered) y adaptación al contexto. Un ejemplo de lo primero sería cuando un usuario entra en una habitación específica, sus mails serían mostrados en una terminal cercana. Un ejemplo de lo segundo sería aumentar el volúmen de un auricular acorde al nivel de ruido actual.

En un sistema sensible al contexto, los dispositivos móviles deberán poseer características especiales. Un dispositivo móvil deberá conocer las preferencias del usuario y, en base a ésta información, facilitarle sus tareas y la interacción del mismo con el entorno, a tal punto de operar sin participación del usuario. Es decir, los dispositivos móviles deberán:

- Percibir el entorno (dispositivo móvil con sensores, puertos bluetooth e infrarojo, cámara, reconocimiento por visión, etc.).
- Almacenar todas las interacciones del usuario (llega al garaje, abre la puerta, llama al ascensor, entra en casa, etc.).
- Extraer patrones de comportamiento (en base a tiempo, lugar, dispositivos vecinos).
- Aplicar este conocimiento según los patrones extraídos (ej: abrir la puerta del garaje cuando se aproxima).

Los dispositivos móviles se convierten en una pieza vital para aplicaciones sensibles al contexto. Esta tesis se centrará en las características de ubicación y actividad del usuario. Dichas características están estrechamente ligadas al funcionamiento del browser Supongamos un usuario que está recorriendo un Museo, algunos ejemplos que podrían darse son en este contexto son:

- El usuario se encuentra caminando y, al cambiar de ubicación, es sensado por un nuevo Objeto Físico, el cuál origina una actualización de la información mostrada en el browser.
- El browser cambia su política de navegación de acuerdo a nuevas necesidades del usuario. Si el usuario se encuentra perdido, la política de navegación física deberá ser más detallada o especifica para poderlo ayudar.
- A medida que el usuario va recorriendo las diferentes salas del Museo, es sensado por las mismas y los links relacionados a las salas se van actualizando. Podría pasar que un usuario se encuentra en la sala A frente a un objeto físico, el usuario podría ver links propios del objeto físico, como así también links relacionados con la sala A. Todos los objetos que se encuentran en la sala A, mostrarán los mismo links asociados a la sala, pero información propia de cada objeto.
- Supongamos que existen, dentro del Museo, salas en las cuáles se ofrecen charlas, conferencias o presentaciones diarias. Cuando un usuario es sensado por alguna de estas salas, se le brinda información sobre próximas conferencias (tema, tipo y horario), dándole la posibilidad de reservar un lugar dentro de ella.

Capítulo 3. Modelo Conceptual de aplicaciones de Hipermedia Física.

3.1. Objetos Digitales y Físicos

Un esquema general que muestra la modelización de los objetos físicos y digitales se puede ver en la Figura 12. Por un lado, están modelados los *PhysicalObjects*, que representan cualquier objeto del mundo real. Como subclase de *PhysicalObject* están los *PointOfInterest* (Objetos de Interés) los cuáles representan lugares de interés para el usuario. *PointOfInterest* conoce a su *DigitalObject* asociado el cual contiene información Hipermedial del punto de interés. Por ejemplo, se podría tener representada La Catedral como un *PointOfInterest,* que tiene asociado su información Hipermedial en el *DigitalObject.* Podría haber DigitalObjects que no fueran conocidos, en forma directa, por ningún punto de interés (por ejemplo: un objeto digital que represente la información del arquitecto que diseño de la Catedral).

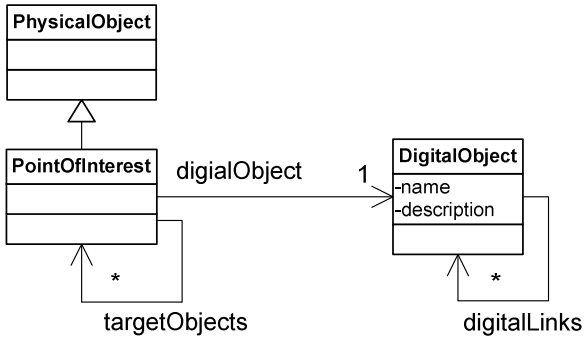


Figura 12: Modelo básico de objetos físicos y digitales.

La relación *digitalObject* representa la contraparte digital del punto de interés. Un *DigitalObject* tendrá un nombre y una descripción (eventualmente se podría tener más datos, solo se muestran estos para simplificar el diagrama). Cuando un usuario es sensado por un objeto *PointOfInterest* muestra la información digital asociada (por ejemplo, el nombre y la descripción). Las otras dos relaciones (*digitalLinks y targetObjects*) que aparecen en la Figura 12, representan el conocimiento digital y físico entre los objetos. En el browser estas relaciones serán interpretadas como links físicos y digitales. Es decir, cuando un usuario es sensado por un *PointOfInterest* tendrá tantos links físicos como objetos físicos haya en *targetObjects* y tantos links digitales como objetos digitales haya en *digitalLinks* (del DigitalObject asociado a ese punto de interés).

Toda esta información es utilizada, por el browser, para mostrar los datos al usuario. La Figura 13 muestra la pantalla con información típica cuando un usuario esta parado en frente de un Objeto de Interés (Objeto Físico) [Fortier et al, 2007]. La página web contendrá 4 categorías principales:

- Información Digital: explican las características de lugar.
- Links Digitales: links de hipermedia tradicional.
- Información Física: acerca de la ubicación del punto de interés.
- Links Físicos: podrán ser utilizados para iniciar una navegación. A partir de estos se genera el camino que debe realizar el usuario.

Figura 13: Información standard de una página Web de PH.

Cuando el usuario clickea sobre un link digital, solo cambia la información digital, mostrando el contenido de la página Web requerida por el usuario, pero la información física no se modifica. En otras palabras, aunque el usuario se encuentre navegando digitalmente, si él aún está frente al mismo punto de interés, la información física debería mantenerse sin cambios. Por otra parte, cuando el usuario clickea en un link físico indica que quiere llegar a un nuevo punto de interés; en ese momento se mostrará un mapa (u otra representación en multimedia) indicando el camino a seguir hasta ese target. Solo cuando el usuario llegue a destino se mostrará la información completa acerca de ese lugar (información y links digitales e información y links físicos). Se tomaran estos conceptos para el desarrollo de la funcionalidad presentada en esta tesis.

Tener estos dos tipos de navegaciones con sus respectivos comportamientos generan la necesidad de tener un modelo genérico de navegación [Challiol et al, 2007], el cual es presentado en la sección 3.3 Dicho modelo es la base para el desarrollo de los modelos de esta tesis, junto con los conceptos presentados en esta sección.

3.2. Estados de un usuario en un sistema de Hipermedia Física

En un sistema de Hipermedia Física tenemos, como actor principal, al usuario, el cuál va navegando físicamente de un objeto real (físico) a otro obteniendo información en su dispositivo móvil. Dentro de este sistema, el usuario tiene asociado un estado (UserState), el cuál puede ser Walking o Navigating.

Si el usuario se encuentra caminando dentro del sistema y no seleccionó navegar hacia un Objeto Físico específico, su estado será Walking. Podemos entender este estado como que el usuario simplemente está recorriendo el ambiente sin un rumbo específico; por el contrario, si el usuario decide iniciar una navegación (clickeando un link físico) a un Objeto Físico su estado será Navigating.

A su vez, el UserState tiene asociada una actividad física (PhysicalActivity); la misma puede ser RequestNavigation, InFrontOf, InPath (dentro del recorrido), OutsidePath (fuera del recorrido), CancelNavigation, ReachTarget o Anywhere.

En caso que el usuario no se encuentre frente a un Objeto Físico (por ejemplo, mientras camina un link físico) su ubicación será representada como Anywhere, el

cuál actúa como un objeto nulo [Fortier et al, 2007]. Representar la actividad Anywhere queda fuera del alcance de esta tesis debido a la complejidad que genera simular este comportamiento dentro del ambiente.

La Figura 14 muestra los posibles estados de un usuario (Walking y Navigating) en un sistema de Hipermedia Física junto a las actividades físicas a llevar a cabo en cada uno (y las posibles transiciones de un estado a otro).

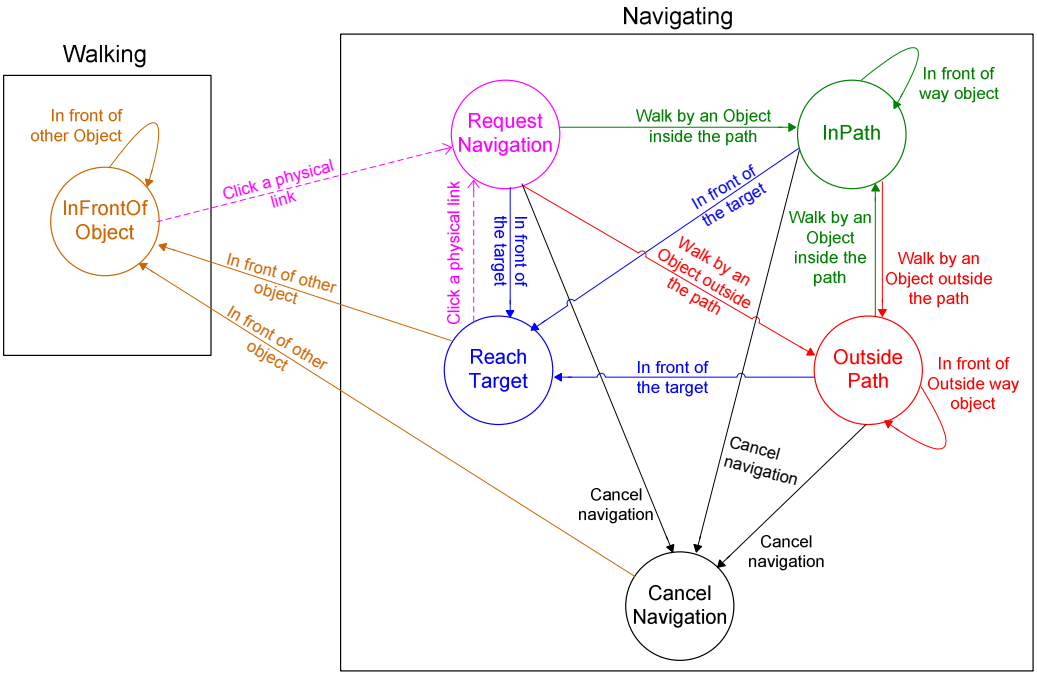


Figura 14: Estados de un usuario y actividades físicas.

Un usuario puede ir de un punto de interés a otro por medio de dos acciones:

- La primera, y la más simple, es que el usuario no indique un destino, simplemente camina por el ambiente siendo sensado por objetos físicos; cuando esto sucede, el UserState del usuario será Walking, y la actividad física será InFrontOf (cuando es sensado).

- El otro caso se da cuando el usuario inició una navegación física explícita (Click en un link físico), en tal caso, el UserState del mismo será Navigating y la actividad física será RequestNavigation. A su vez, cuando un usuario está dentro de una navegación (Navigating), puede estar, entre otras, dentro o fuera del recorrido sugerido por el sistema, en caso de encontrarse dentro del recorrido sugerido la actividad del usuario será InPath; si se encuentra fuera del recorrido sugerido su actividad física será OutsidePath. Cuando un usuario cancele una navegación su UserState pasará a ser Walking y su PhysicalActivity, InFrontOf; en el caso de llegar al destino solicitado, el UserState será Navigating con la actividad física ReachTarget.

Estos estados contribuyen a una mejor comprensión de la semántica de una navegación física con el fin de poder desarrollar una política de navegación acorde al comportamiento del usuario. El comportamiento del usuario va a ser considerado a la hora de desarrollar las políticas físicas sobre la base del modelo que será presentado en el siguiente capítulo.

3.3. Modelo de navegación Abstracto

El modelo definido en [Challiol et al, 2007] sirvió como base para el desarrollo de esta tesis. Este modelo básico especifica los conceptos más generales para navegar dominios que tengan un esquema en el cual se quiera navegar haciendo back y next. En la Figura 15 se presenta un modelo básico de un Browser (BasicBrowserModel). Este modelo sirvió de base tanto para desarrollar el Browser Digital (con su correspondiente navegación digital) así como también para el Browser Físico (con la navegación física correspondiente).

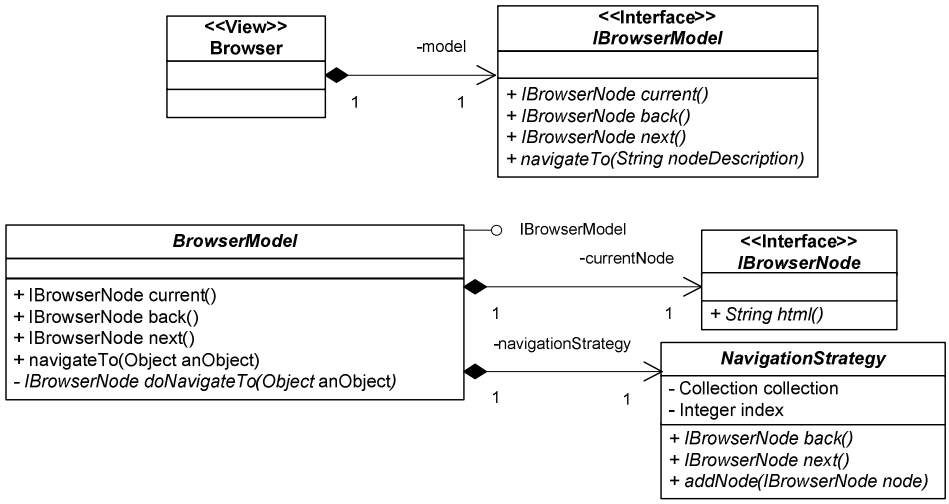


Figura 15: BasicBrowserModel [Challiol et al, 2007]

Veamos a continuación los componentes descriptos por el modelo básico presentado en la Figura 15. *Browser* representa la vista de un Browser, este conoce un modelo el cual debe implementar la interfaz *IBrowserModel*. Es decir, un *Browser* siempre interactuara con objetos que implementen dicha interfaz para poder mostrar nodos en la vista.

La interfaz *IBrowserModel* define el siguiente protocolo (el cual permitirá que el *Browser* interactúe a través de dicho protocolo):

- `current`, retorna el nodo actual asociado al Browser. Es decir, retorna el nodo que esta viendo el usuario. Dicho nodo debe implementar la interfaz *IBrowserNode.*
- `back`, retorna el nodo anteriormente visitado (el nodo debe implementar la interfaz *IBrowserNode*),
- `next`, retorna el nodo siguiente al que esta viendo, el nodo debe implementar la interfaz *IBrowserNode*.

- `navigateTo:anObject`, a partir de `anObject` se generará el próximo nodo que vera el usuario. Eventualmente podrá ser una descripción, la cual permitirá que se busque entre los nodos conocidos por el nodo actual, el nodo que coincide con la descripción (este es el próximo nodo que vera el usuario). Pero cada subclase podría recibir objetos de clases particulares (subclases de `Object`) según su naturaleza.

BrowserModel es una clase abstracta que implementa la interfaz *IBrowserModel* descripta anteriormente. *BrowserModel* conoce su `currentNode` (objeto que implementa la interfaz *IBrowserNode*), representa el nodo actual que visualiza el *Browser.* Además, *BrowserModel* conoce la estrategia de navegación (*NavigationStrategy*), esta estrategia permitirá poder implementar distintos algoritmos (patrón Strategy [Gamma, 1995]) de backtraking.

IBrowserNode define un comportamiento común para cualquier nodo. El *IBrowserNode* especifica que la clase que implemente dicha interafaz tendrá que definir el método `html`. El método `html` devuelve un *String* que representa el HTML asociado al nodo.

En cuanto a *NavigationStrategy* representa la política o estrategia de navegación subyacente de un Browser. La política de navegación será independiente del Browser. El *BrowserModel* conoce a su política a través de `navegationStrategy`. *NavigationStrategy* es una clase abstracta que sirve para definir las nuevas políticas de navegación. Dentro de esta se definen 3 métodos abstractos `back`, `next` y `addNode:aBrowserNode`. `Back` y `next` son los métodos encargados de retornar nodos anteriores o posteriores a la posición actual mientras que `addNode:aBrowserNode` será el método encargado de agregar un nuevo nodo a la lista de nodos visitados. Al ser estos métodos abstractos, la implentación de los mismos deberá ser realizada por las nuevas subclases (nuevas políticas de navegación). *NavigationStrategy* tiene `collection` donde guarda todos los nodos visitados, e `index` que indica una posición dentro de la colección de nodos visitados.

Retomemos con la clase *BrowserModel* y veamos como implementa la interfaz *IBrowserModel* y que métodos nuevos implementa. Los métodos que implementa la interfaz se pueden visualizar en el Código 1, 2 y 3 (el método `current` devuelve directamente el `currentNode`, tomando de `collection` el elemento en la posición indicada por `index`, por tal motivo no se especificara):

```
back
     self currentNode: self navigationModel back.
     ^self currentNode.
```

Código 1: Método `back` de BrowserModel.

```
next
     self currentNode: self navigationModel next.
     ^self currentNode.
```

Código 2: Método `next` de BrowserModel.

Se puede apreciar que el back y next delegan en el modelo de navegación la resolución del back y next (ver Código 1 y 2), seteando lo que devuelve como nodo actual y además lo retornan.
En el Código 3, se visualiza la llamada a un nuevo mensaje `doNavigateTo: anObject`, sirve para determinar cual será el nodo actual a partir de `anObject` que llega como parámetro. Además se le indica al modelo de navegación que debe agregar el nuevo nodo a los nodos ya visitados. El método `navigateTo: anObject` es el punto inicial para comenzar una navegación.

```
navigateTo: anObject
     self currentNode: (self doNavigateTo: anObject).
     self navigationModel addNode: self currentNode.
     ^self currentNode.
```

Código 3: Método navigateTo(String) de BrowserModel.

El método `doNavigateTo:anObject`, deberá ser implementado por todas las subclases de *BrowserModel.* Este método toma un `Object` y deberá retornar un nodo (el cual implementa la interfaz *IBrowserNode)*. Es decir, *BrowserModel* define este método como abstracto. Como cada subclase de *BrowserModel* define este método, cada una manipulara el objeto de acuerdo a su naturaleza. Por ejemplo: podría llegar un String que identifica el objeto y en este método se deberá buscar el objeto asociado. O podría llegar directamente el objeto al cual se quiere navegar y este método simplemente construirá el nodo asociado.

La Figura 16, muestra un diagrama de secuencia general que será compartido por todos los modelos y estrategias de navegación. En negrita se destaca los métodos que deben ser implementados por las subclases *BrowserModel* y *NavigationStrategy.*

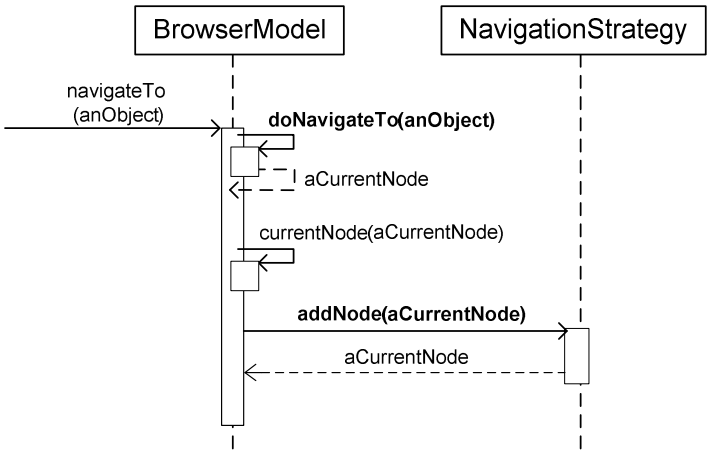


Figura 16: Diagrama de secuencia abstracto de `navigateTo:anObject`

Finalmente, la clase *BrowserModel* no solo provee una implementación para la interfaz *IBrowserModel*, sino que también permite desacopla la funcionalidad de Back y Next delegándola en la estrategia de navegación.

El modelo básico presentado en esta sección fue utilizado tanto a nivel conceptual como para extenderlo y poder desarrollar el modelo digital y físicos presentados esta tesis). Estos modelos además se construyen sobre los conceptos presentados en la sección 3.1 y 3.2, los mismos serán presentados en el siguiente capitulo.

Capítulo 4. Browser para Hipermedia Física.

Para llevar a cabo este desarrollo se utilizó VisualWorks 7.4.1, un ambiente de desarrollo Smalltalk robusto multi-plataforma. Todas las clases junto con los métodos que se mencionan en esta sección fueron desarrolladas en este ambiente.

Dada la complejidad que requiere desarrollar la funcionalidad de un Browser para Hipermedia Física, se dividió la puesta en práctica de esta funcionalidad en dos grandes etapas, una primera relacionada con la navegación digital y una segunda que involucra todo lo relacionado con la navegación física.

En la primera etapa, se estudiaron la semántica y las políticas de navegación existentes en los navegadores Web tradicionales. Este estudio permitió comprender y aplicar estos conceptos al desarrollo de la funcionalidad del Browser en la parte de navegación digital.

La segunda etapa, navegación física, requirió de mayor análisis y estudio debido a la escasa documentación asociada al tema, por lo reciente de estas aplicaciones; una vez comprendida la semántica relacionada con la navegación física se diseñaron nuevas políticas para dicha navegación.

En las siguientes secciones se verá como fueron puestas en práctica estas dos grandes etapas.

4.1. Navegación Digital.

4.1.1. Semántica de navegación la navegación digital.

La semántica de navegación digital para un Browser de Hipermedia Física es la misma que utilizan los Browser Web tradicional. Como se describió en el capítulo 2.3.1., la funcionalidad básica de un navegador Web es permitir la visualización de páginas Webs. Tales páginas, poseen hipervínculos que enlazan una porción de texto o imagen a otra página. El seguimiento de enlaces (links) de una página a otra, ubicada en cualquier computadora conectada a Internet, se llama navegación (cuando hablamos de navegación en un Browser de Hipermedia tradicional nos referimos a navegación digital). Un Browser posee botones standard; entre los más comunes se encuentran los botones de Back, Next, Reload y Stop.

4.1.2. Modelo de navegación digital

En esta sección se presentara como a partir del BasicBrowserModel (presentado en el sección 3.3), se definió el modelo para un Browser Digital (DigitalBrowserModel). En la Figura 17 se presenta el diagrama del DigitalBrowserModel. En este diagrama surge una nueva clase *DigitalNode* para representar los nodos digitales (dicha clase implementa la interfaz *IBrowserNode*). Un *DigitalNode* actúa como Wrapper [Gamma, 1995] el cuál agrega información al objeto wrapeado (objeto digital, concepto presentado en la sección 3.1), y le

agrega todo lo relacionado a HTML. Un *DigitalNode* posee dos variables de instancia, `target` y `path`. `Target` representa el objeto digital de ese *DigitalNode* y `path` tiene asociado la ruta para ubicar el template que permite la visualización en HTML. El template es utilizado por el método `html` para a través de los tags del template extraer la información del *DigitalNode,* y generar el HTML asociado.

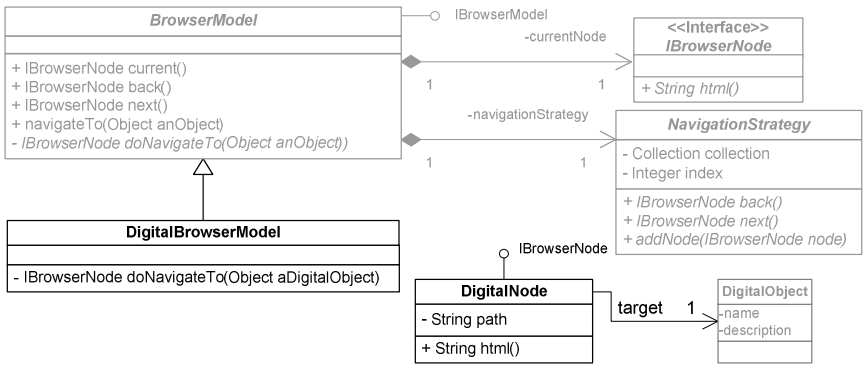


Figura 17: Diagrama del DigitalBrowserModel.

En la Figura 17, también se puede apreciar que se extendió *BrowserModel*, creando como subclase *DigitalBrowserModel*, esta clase redefine el método `doNavigateTo:anObject`. En el Código 4 se muestra el código de dicho método, donde `anObject` es un *DigitalObject* (objeto digital al cual se quiere navegar).

```
doNavigateTo: aDigitalObject
  ^DigitalNode for: aDigitalObject path: self path.
```

Código 4: Método `doNavigateTo:Object` de DigitalBrowserModel.

En el Código 4 se crea un nuevo Nodo Digital utilizando la información de *aDigitalObject* y el path donde se encuentra el template que permite visualizar HTML.

Veamos como seria la instanciación de las clases presentadas en la Figura 16 y Figura 17, para tener un *DigitalBrowserModel* cuando el usuario esta viendo un *DigitalNode* (el cual contiene un objeto digital,"Catedral").

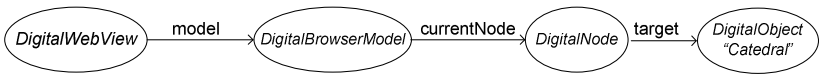


Figura 18: Diagrama de instanciación de un DigitalBrowserModel.

En la Figura 18 se puede apreciar que el *DigitalWebView* (que representa un Browser Digital) conocerá a un objeto de la clase *DigitalBrowserModel* (modelo digital subyacente al Browser). A su vez el *DigitalBrowserModel* conocerá, como nodo actual, a un objeto de la clase *DigitalNode*, el cuál por medio de su `target` conocerá su *DigitalObject* (en este ejemplo, "Catedral").

La actualización del `currentNode` de un *DigitalBrowserModel* se origina con una nueva navegación digital tradicional (como en la Web), ya sea porque el usuario clickeó sobre un nuevo link (El Browser delegara en su `model` la resolución). El

Browser envía el mensaje `navigateTo:anObject` a su `model` (en el caso digital al *DigitalBrowserModel*). También se puede actualizar porque decidió revisitar un Nodo por medio de los botones Back o Next (El Browser delegara en su `model` la resolución, como se mostrara más adelante).

Existe otro modo de actualizar el `currentNode` y se da cuando un usuario es sensado por un nuevo punto de interés (esto se vera más adelante en la sección 4.2.3).

Hasta el momento se hizo mención al `currentNode` asociado al *DigitalBrowserModel,* en la siguiente sección se hará mención a las políticas de navegación digitales asociadas a dicho modelo.

4.1.3. Políticas de navegación digitales

En el capítulo 2.4.2 se detallaron dos políticas de navegación digital, Stack-Based y Recency-Based. Estas dos políticas son las que se desarrollarán en esta tesis. Cabe aclarar que existen otras políticas de navegación actuales que no serán desarrolladas en esta tesis ([Cockburn et al, 2002], [Milic-Frayling et al, 2004] se puede encontrar más información al respecto).

A partir de *NavigationStrategy* (Figura 15, sección 3.3) se crearan las distintas políticas como subclase de esta clase. Las subclases de *NavigationStrategy* deberán definir los métodos: `back`, `next` y `addNode:aBrowserNode`. `Back` y `next` son los métodos encargados de retornar nodos anteriores o posteriores a la posición actual mientras que `addNode:aBrowserNode` será el método encargado de agregar un nuevo nodo a la lista de nodos visitados. Cada política implementa estos métodos en base a la semántica de su funcionamiento.

4.1.3.1. Modelo Stack-Based.

Partiendo del diagrama presentado en la Figura 15 (sección 3.3), se creó la clase *StackBasedStrategy* (como subclase de *NavigationStrategy*). *StackBasedStrategy* representara la estrategia de navegación Stack-Based presentada en el punto 2.4.2 de esta tesis. La Figura 19, muestra como se incorpora la nueva clase StackBasedStrategy.

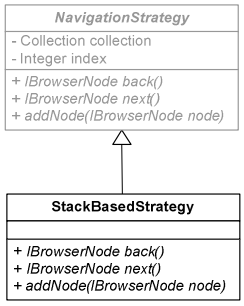


Figura 19: Extensión del modelo con la política StackBasedStrategy.

A continuación se muestran los métodos definidos por la política StackBasedStrategy.

```
back
     (self index > 1)
        ifTrue:[self index: self index - 1].
     ^self currentNode.
```

Código 5: Método `back` de StackBasedStrategy.

El Código 5 implementa un retroceso en el historial de navegación retornando un nuevo nodo actual (El `currentNode` se calcula a partir de `collection`, el elemento que se encuentra en la posición que indica el `index`); siempre y cuando el anterior nodo no fuere el primer nodo de la colección.

De manera análoga, el Código 6 implementa un avance en el historial de navegación retornando el nuevo nodo actual; siempre que el anterior nodo no sea el último de la colección.

```
next
     (self index < self collection size)
        ifTrue:[self index: self index + 1].
     ^self currentNode.
```

Código 6: Método `next` de StackBasedStrategy.

El Código 7 muestra como un nodo es agregado al historial de navegación de la política. Al inicio de éste método se ve como, si el índice es distinto del tamaño de la colección, se descartan los nodos que estén a continuación de la posición indicada por `index`. Se pierden nodos visitados cuando el usuario navega con el back y next y se clickea a un link (comportamiento particular de esta política explicado en la sección 2.4.2).

```
addNode: aBrowserNode
     (self index ~= self collection size)
        ifTrue:[self collection: (self collection copyFrom: 1 to:
        self index)].
     self collection add: aBrowserNode.
     self index: self index + 1.
     ^self currentNode.
```

Código 7: Método `addNode:aBrowserNode` de StackBasedStrategy.

En la Figura 20, se puede apreciar que el *DigitalWebView* (que representa un Browser Digital) conocerá a un objeto de la clase *DigitalBrowserModel* (representando el modelo digital subyacente al Browser). A su vez el *DigitalBrowserModel* conocerá su política de navegación, en este caso, *StackBasedStrategy*.

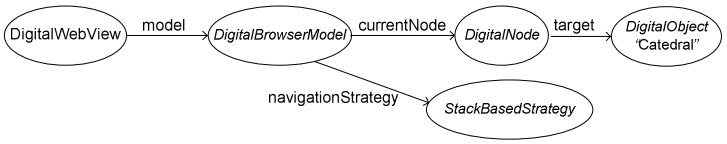


Figura 20: Diagrama de instanciación con una política StackBasedStrategy.

La Figura 21 muestra el diagrama de secuencia que se genera cuando el usuario "clickea un link" con política Stack-Based:

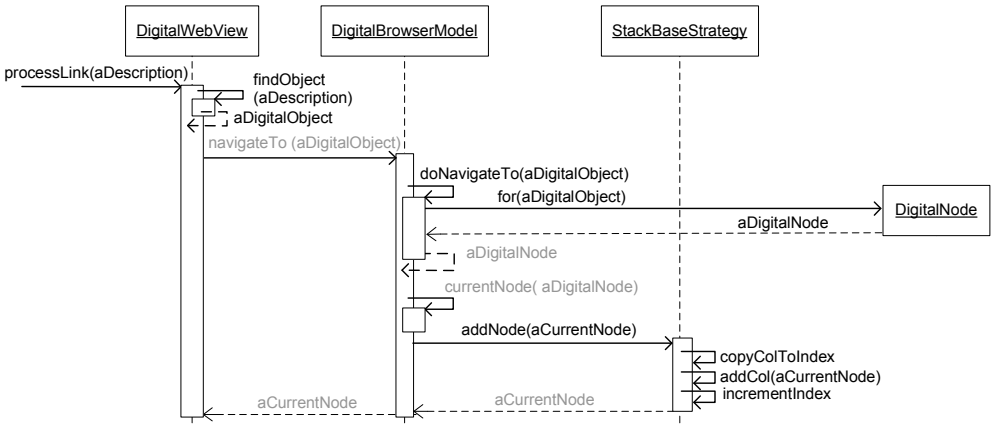


Figura 21: Diagrama de secuencia generado al clickear un link.

En la Figura 21, el mensaje `processLink` es el mensaje inicial que le llega al DigitalWebView (ventana de un Browser Digital) cuando el usuario clickea un link digital. Con `findObject:aDescription` se logra encontrar el objeto digital al cual se quiere navegar a partir de la descripción del mismo. Luego se puede ver en la Figura 21 que se sigue la secuencia de mensajes de acuerdo al Código 3 (en gris se representa la estructura heredada). Se puede apreciar que *DigitalWebView* delega en *DigitalBrowserModel* la resolución del link y recibe el nuevo *DigitalNode* que debe mostrar.

La Figura 22 muestra el diagrama de secuencia que se genera cuando el usuario decide hacer Back (análogo a Next) con política Stack-Based:

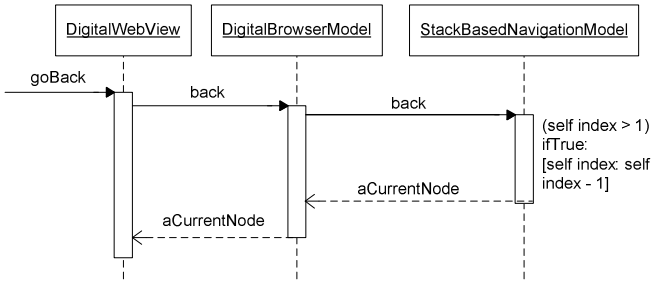


Figura 22: Diagrama de secuencia generado al clickear un link.

En la Figura 22, el mensaje `goBack` es el mensaje inicial que le llega al *DigitalWebView* (ventana de un Browser Digital) cuando el usuario oprime el botón de Back digital. Se puede ver en la Figura 22 que se sigue la secuencia de mensajes de acuerdo al Código 1. Nuevamente se puede apreciar que *DigitalWebView* delega en *DigitalBrowserModel* la resolución del Back y recibe el nuevo *DigitalNode* que debe mostrar. A partir de esto se puede apreciar como el *DigitalWebView* (Browser) se independiza del modelo navegacional.

4.1.3.2. Modelo Recency-Based.

Para lograr agregar una nueva política de navegación al modelo existente se crea la clase *RecencyBasedStrategy* como subclase de la clase *NavigationStrategy*, (como se puede apreciar en la Figura 23). Esta nueva clase implementa la política Recency-Based descripta en el punto 2.4.2.

Como se mencionó anteriormente, la política de navegación es independiente de la vista del Browser. Agregar una nueva política no altera el comportamiento de *DigitalBrowserModel* (en un momento dado *DigitalBrowserModel* conocerá a una sola política). En cuanto a funcionalidad, una nueva política de navegación aporta una nueva manera de revisitar nodos, por medio del Back y Next. Una nueva política solo necesita redefinir los métodos indicados en la clase abstracta *NavigationStrategy* (como se mostró para *StackBasedStrategy*).

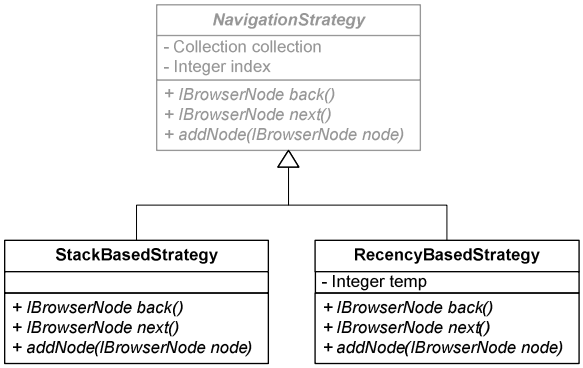


Figura 23: Incorporación de la política RecencyBasedStrategy.

A continuación se muestran como implementa en *RecencyBasedStrategy* los métodos `back, next` y `addNode:aBrowserNode`:

```
back
     self addTemp: self currentNode.
     (self index > 1) ifTrue:[ self index: self index - 1 ].
     ^self currentNode.
```

Código 8: Método `back` de RecencyBasedStrategy.

El Código 8 implementa un retroceso en el historial de navegación retornando un nuevo nodo actual (`^self currentNode`); siempre y cuando el anterior nodo actual no fuere el primer nodo de la colección. Además, se guarda en el `temp` el `currentNode`.
De manera análoga, el Código 9 implementa un avance en el historial de navegación retornando el nuevo nodo actual; siempre que el anterior nodo actual no sea el último de la colección. Y se guarda en el `temp` el `currentNode`.

```
next
     self addTemp: self currentNode.
     (self index < self collection size) ifTrue:[ self index:
     self index + 1 ].
     ^self currentNode.
```

Código 9: Método `next` de RecencyBasedStrategy.

A diferencia de los códigos 5 y 6 (`back` y `next` de *StackBasedStrategy* sección 4.1.3.1), se ve como en los Códigos 8 y 9 se guarda, antes de calcular el nuevo `index` (que servirá para determinar el nuevo nodo actual), el `currentNode` en una lista temporal; esto lo hace para luego poder reordenar la lista con todos los nodos visitados. Esto forma parte de la semántica de la política Recency-Based enunciada en 2.4.2.

El Código 10 muestra como un nodo es agregado al historial de navegación. Como se puede ver, agregar un nodo en una política RecencyBasedStrategy resulta más complejo que hacerlo en una StackBasedStrategy.

```
addNode: aBrowserNode
     (self index > 0) ifTrue:[self addTemp: self currentNode].
     self collection: ( self temp reverse ) copy.
     self deleteBrowserNode: aBrowserNode.
     self collection add: aBrowserNode.
     self index: (self collection size).
     ^self currentNode.
```

Código 10: Método `addNode:IBrowserNode` **de RecencyBasedStrategy.**

Se puede observar en el Código 10, para llevar a cabo dicha operación se deben seguir algunos pasos: en primer lugar, si `collection` no se encuentra vacía, se agrega `currentNode` en `temp` (ver Código 11), borrando los duplicados dentro de la misma. Posterior a ésto, `temp` se copia en el historial de navegación (`collection`) con el objetivo de tener dentro de éste, todos los nodos visitados ordenados por más reciente. Una vez copiado `temp` en `collection`, se borran todas las ocurrencias de `aBrowserNode` dentro de `collection`, con el objetivo de eliminar los posibles duplicados, para luego agregarlo al tope de la misma; al final, se actualiza el índice `index` quedando `aBrowserNode` como `current`.

Con el fin de aportar mayor claridad a los códigos mencionados en esta sección, a continuación se presentarán algunos métodos privados de *RecencyBasedStrategy*: En el Código 11 se ve como se agrega un nodo al principio de la lista y posteriormente se borran los duplicados (ver Código 12) dentro de la misma.

```
addTemp: aCurrentNode
     self temp addFirst: aCurrentNode.
     self deleteDuplicated.
```

Código 11: Método `addTemp:IBrowserNode` **de RecencyBasedStrategy.**

En el Código 12 se muestra el borrado real de esos duplicados (ya que de esta manera funciona la política). Internamente, lo primero que hace este código es detectar (y copiar en una lista auxiliar `aux`) todos los objetos sin incluir duplicados, luego vacía `temp` y por último, copia todos los objetos de `aux` en `temp`.

```
deleteDuplicated
     | aux |
     aux := OrderedCollection new.
     (self temp copy) do: [ :o | aux detect: [ :a | a target = o
     target ]
          ifNone:[ aux add: o ]  ].
```

```
    self temp removeAll: self temp copy.
    self temp addAll: aux.
```

Código 12: Método `deleteDuplicated` de RecencyBasedStrategy.

El Código 13 muestra como se borran todas las ocurrencias de un *BrowserNode* dentro de la lista de nodos visitados. El método `removeAllSuchThat:` remueve el objeto si el bloque evaluado resulta verdadero.

```
deleteBrowserNode: aBrowserNode
    self collection removeAllSuchThat:[ :a | a
    target=aBrowserNode target. ].
```

Código 13: Método `deleteBrowserNode:IBrowserNode` de RecencyBasedStrategy.

La Figura 24 se muestra una instancia donde *DigitalBrowserModel* ahora conocerá como política de navegación, a *RecencyBasedStrategy*.

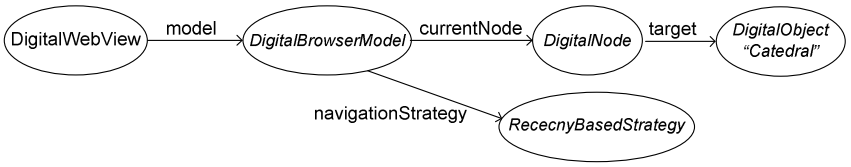


Figura 24: Diagrama de instanciación de una política RecencyBasedStrategy.

La Figura 25 muestra el modelo donde se incluye *DigitalBrowserModel*, *DigitalNode* y las dos políticas de navegación digital Stack-Based y Recency-Based.

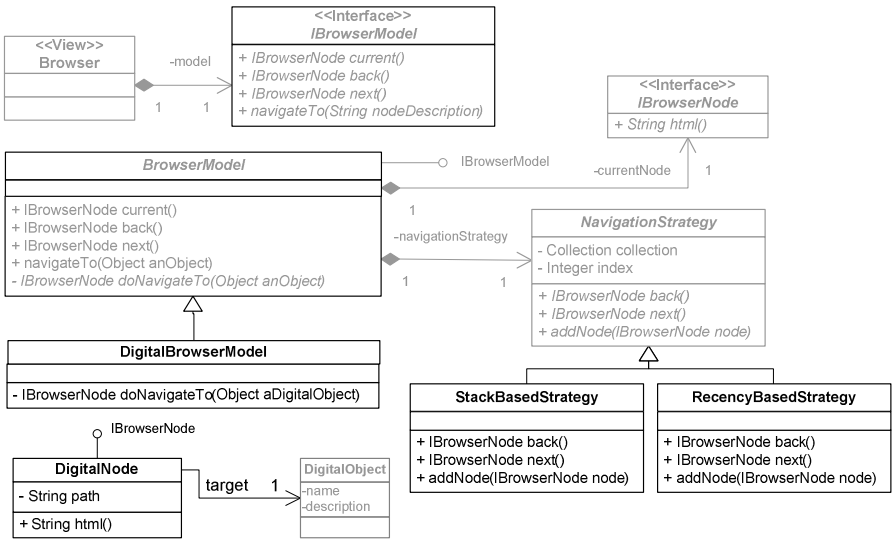


Figura 25: DigitalBrowserModel con distintas estrategias de navegación Digital.

En resumen, a lo largo de este capítulo se presentó el modelo de un Browser Digital junto con dos posibles políticas de navegación digital (Stack-Based y Recency-Based). En la siguiente sección se presentara el modelo definido para el Browser Físico.

4.2. Navegación Física.

4.2.1. Semántica de una navegación física.

La semántica de una navegación física requiere un análisis más profundo debido a la mayor complejidad que ésta presenta respecto de una navegación digital. La complejidad se genera al involucrar el movimiento del usuario en el mundo real en esta navegación. Es decir, la navegación física no es atómica desde el punto de vista del usuario como lo es la navegación digital, sino que involucra que el usuario cambie de posición.

Una navegación física consiste en ir desde un punto de interés origen a un punto de interés destino (target). Cuando se inicia una navegación física (un usuario manifiesta su intención de obtener información del target clickeando en un link físico), se recibe información que detalla como llegar hasta el target del link. Obviamente, el sistema no puede mover el target, por eso su objetivo en este tipo de navegación (física) es asistir al usuario para poder alcanzar el target seleccionado. Mientras el usuario recorre el camino sugerido, se puede cruzar con otros objetos físicos los cuáles le aportarán información adicional ayudándolo a completar su recorrido físico [Challiol et al, 2008]. Es decir, el usuario debe caminar hasta el target para visualizar su información, esto se conoce como "caminar un link" ("Walk the link", [Goble et al, 2004]). Cuando el usuario "camina un link" lo que está haciendo es realizar una navegación física. Una navegación de este tipo implica que el usuario seleccione un link físico (por medio de un Click en link), y camine, siguiendo el recorrido sugerido por el sistema, para alcanzar el objeto target; cuando el usuario llega al objeto target recién en ese momento la navegación física se considera finalizada.

Con motivo de ubicarnos en un ambiente más conocido se podría establecer una cierta similitud entre ambas navegaciones (física y digital), en sentido que, en ambos casos existen las mismas acciones: Click en un link, Back, Next, Reload y Stop. Como se vió en 4.3.2, existen ciertas similitudes entre navegacion física y digital; a continuación se enuncian las semánticas de Back, Next, Reload y Stop para la parte física:

- Back: inicia una nueva navegación física a un objeto anterior a la posición actual.
- Next: inicia una nueva navegación física a un objeto siguiente a la posición actual.
- Reload: se actualiza el nodo físico que está viendo; esto implica recargar, si existe, la contraparte digital del punto de interés actual. En caso de ser un objeto físico puro (no tiene contraparte digital), el contenido del *Browser Digital* no se modificará.
- Stop: cancela la navegación física a ese nodo target, por lo tanto se cancela el recorrido actual.

Usualmente, un usuario se va moviendo de un objeto a otro obteniendo información pero que sucedería si, en un determinado momento, el usuario decide retroceder a un punto de interés que había visitado? El usuario tendría que tener la misma posibilidad de recorrer su historial de navegación física (como el Browser lo

permite en el caso de la navegación digital), pudiendo retroceder a objetos ya visitados. Esta idea muestra que retroceder a un objeto visitado (hacer Back) implica moverse de un objeto a otro; razón por la cuál un retroceso físico genera mayor dificultad.

A continuación se muestra, con un ejemplo, la semántica del Back físico (análogo para el Next). Supongamos que tenemos Objetos físicos distribuidos de la siguiente manera (ver Figura 26, las relaciones son links físicos):

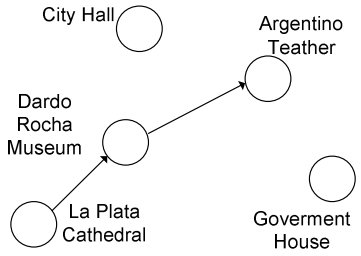
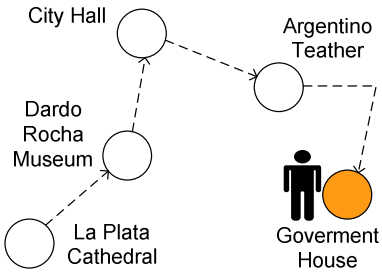


Figura 26: Mapa de Objetos Físicos.

Inicialmente, el usuario está parado frente a “La Plata Catedral”. Pudiendo visualizar una pantalla similar a la presentada en el punto 3.1 (Figura 13). El usuario visualiza un link físico “Dardo Rocha Museum”, decide seleccionarlo (es decir, expresa la intención de conocer ese lugar). El sistema le presenta un mapa, el usuario siguiendo el mismo llega al “Dardo Rocha Museum”. Visualizando la información de “Dardo Rocha Museum” ve un link físico a “Argentino Teather”, lo selecciona y visualiza el camino. Mientras va caminando a “Argentino Teather” es sensado por el “City Hall”, unos minutos mas tarde llega a “Argentino Teather”. Luego sigue caminando sin rumbo hasta ser sensado por “Goverment House”, y se queda parado mirando la información de este lugar. En la Figura 27 se puede visualizar como fue quedando el historial de navegación y el recorrido. El historial de navegación se representará como una colección de nodos y un puntero, el cuál indicará el nodo actual. Entre los nodos se pueden visualizar aquellos que son accedidos porque el usuario fue sensado (mediante el nombre textual del punto de interés) y aquellos nodos producto de clickear un link físico (Nav To *NombreObjeto*). Es decir, en el Historial de la Figura 27 se tiene que se agregaron dos nodos producto de clickear un link físico los cuales fueron para querer llegar a “Dardo Rocha Museum” y “Argentino Teather” respectivamente. El resto fue ingresado al historial producto de que el usuario fue sensado por los objetos físicos.



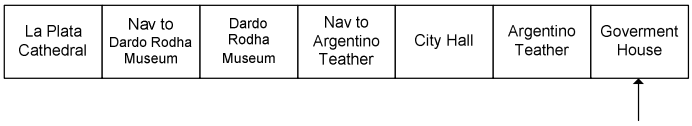


Figura 27: Historial de una navegación física.

Supongamos que en un momento determinado el usuario decide retornar a objetos ya visitados porque se dio cuenta que seria bueno comprar un presente para llevar a su familia. El usuario selecciona un Back físico, por lo tanto se le muestra un mapa junto con el objeto anterior en el que había estado, va a retroceder por los objetos por los que pasó tantas veces como veces seleccione el botón Back, con el Next adelanta sobre esos objetos. Es decir, los botones Back y Next los utiliza para elegir el nodo a revisitar. Esto puede visualizarse en la Figura 28, donde el usuario esta parado en “Goverment House”, y usa el back y next para elegir a donde quiere retornar (se marca con rojo el circulo).

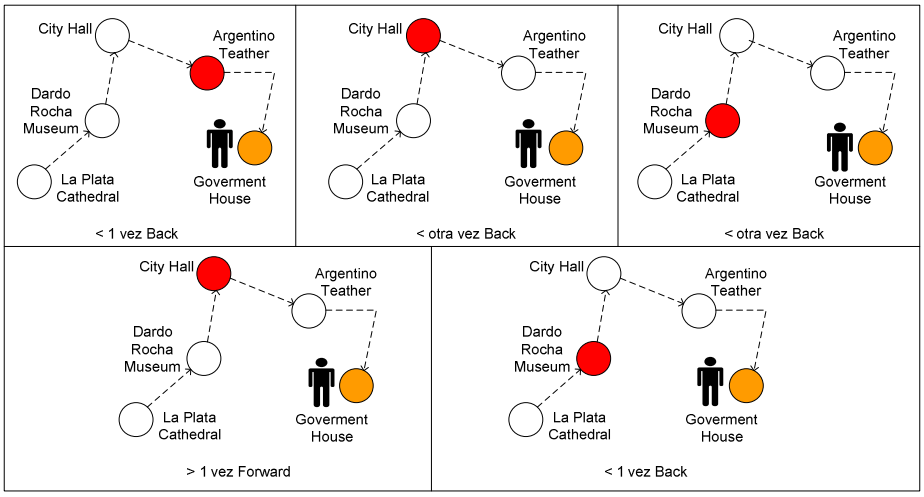
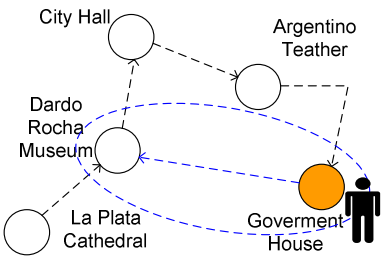


Figura 28: Mapa de posibles back y next físicos.

Luego de hacer back o next decidiendo donde era que había visto el presente que quería comprar, se decide por volver al “Dardo Rocha Museum”. Es decir, elige un target a donde quiere retornar y se le mostrará un camino a recorrer por el usuario. No se completará el back hasta que llegue al objeto al cual quiso retornar. Por lo tanto, por las características que tiene el back (elegir un target, caminar y alcanzarlo), se puede ver como una nueva navegación. Esto se puede visualizar en la Figura 29 (se agregó una nueva navegación y se cambio el puntero). Se generó una nueva navegación en la lista y el puntero esta en “Dardo Rocha Museum”, ya que es la referencia para hacer Back y Next a partir de allí.



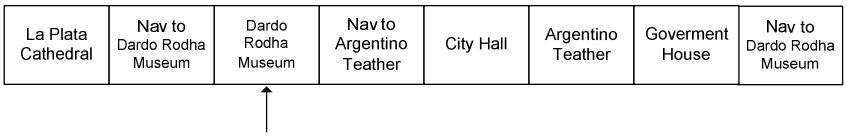


Figura 29: Historial de una navegación física – nueva navegación.

Luego de caminar un rato, finalmente llega a "Dardo Rocha Museum". Pudiendo comprar lo que había querido. En ese momento se cambia el puntero de lugar en la navegación y se agrega "Dardo Rocha Museum" como ya alcanzado. Funciona de manera similar a la navegación generada a través de un link. Esto se puede visualizar en la Figura 30.

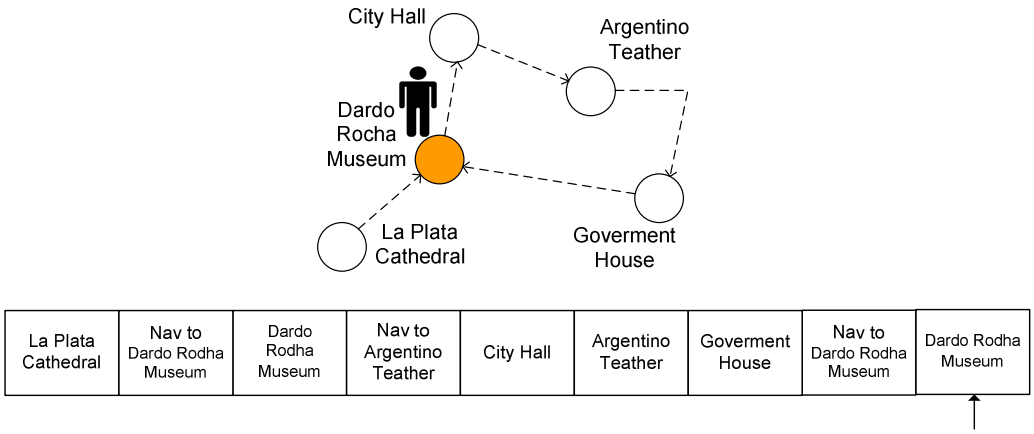


Figura 30: Historial de una navegación física – destino alcanzado.

El puntero que permite hacer referencia vuelve al principio ya que como se realizo una nueva navegación, el Back y el Next es relativo a este nuevo Objeto (Similar a Hipermedia tradicional cuando se navega a una url distinta de las que estaban en el Back y Next). Ahora el Back del "Dardo Rocha Museum" es "Goverment House", ya que ahí estuvo antes de llegar a "Dardo Rocha Museum". Es decir, como el back físico implica una nueva navegación real para llegar a ese punto de interés, una vez que llegó el Back y Next se actualizan en base a ese nodo. Contrario al Back y Next digital, el usuario va a una página y desde allí tiene el Back y el Next en relación a donde había navegado en ese momento. El concepto de caminar y moverse hace que cambie la semántica del back y Next. No va ha haber Next real (caminando) cuando el usuario esta parado en un determinado lugar, ya que desde allí solo tiene lugares que recorrió (es decir, solo hace Back real de recorridos).

Hasta el momento se planteo la semántica de una navegación Física, en la sección 4.2.3 se presentará como se extendió el modelo presentado en la sección 3.3, incorporando los conceptos físicos junto con la navegación física. En la siguiente sección se presentara la representación de los nodos físicos.

4.2.2. Nodos en un modelo de navegación físico.

En la parte digital se vio como un *DigitalBrowserModel* (4.1.2 - Figura 17)

representa, por medio de los *DigitalNode*, sus nodos digitales. Como esos nodos digitales a mostrar en pantalla son, en esencia, idénticos (lo único que cambia entre nodos es la información que describe a cada uno) alcanza con tener una sola clase para representarlos.

Por otro lado, dentro de una navegación física, existen diferentes actividades o estadías en las cuáles un usuario se puede encontrar. En cada una de esas actividades se tendría que mostrar en pantalla información al respecto. Por ejemplo, cuando el usuario inicie una nueva navegación se mostrará en pantalla información respecto a la nueva navegación, como ser objeto origen, objeto destino y camino sugerido; por otro lado, si el usuario simplemente se mueve en el ambiente, no va a ser necesario mostrar en pantalla los datos de una navegación sino simplemente mostrar que objeto lo sensó y a donde puede ir, a partir de ese punto de interés (sus links físicos). Estas ideas derivan en la necesidad de tener diferentes nodos para representar información acorde a las diferentes actividades en las cuáles se pueda encontrar un usuario.

Para poder representar lo antes enunciado se armó una jerarquía de clases partiendo de *PhysicalNode*. La idea de la misma es poder mostrarle al usuario según su actividad (sección 3.2), información acorde a la tarea que esta realizando. Para esto hay que tener nodos específicos como se puede ver en la Figura 31. Algunos de estos serán utilizados en la sección 4.2.3 en los distintos métodos.

Como se ve en la Figura 31, la jerarquía de *PhysicalNode* se divide, a simple vista, en dos ramas, por un lado tenemos *PhysicalObjectNode* y por el otro todo lo relacionado a una navegación, *TravelNode* y sus subclases. Cada nodo mostrara información particular.

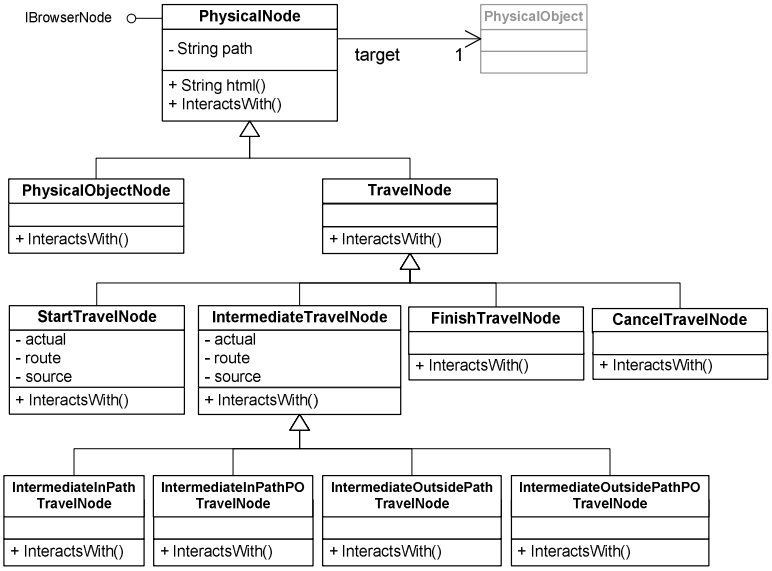


Figura 31: Jerarquía de clase básica de PhysicalNode.

PhysicalNode define el método `html`, el mismo permitirá la visualización en pantalla de los nodos a través de templates. Cada nodo tendrá su template

correspondiente con el objetivo de poder mostrar en pantalla información acorde al mismo. En 'Anexo A: Templates' se verá este tema en profundidad.

El *PhysicalObjectNode* es utilizado por *PhysicalBrowserModel* para instanciar un nuevo nodo cuando el usuario es sensado por un punto de interés y éste se encuentra caminado (userState Walking); este nuevo nodo será su `currentNode`.

A continuación se exhibirán ejemplos de instanciación de subclases de *TravelNode*. El caso inicial será cuando el usuario decide visitar un punto de interés por medio de una navegación explícita (click en un link físico), en ese momento se instancia un nuevo *StartTravelNode* el cuál mostrará en pantalla el objeto actual (origen), objeto destino y recorrido sugerido (objetos intermedios por los que debería pasar). Éste es el punto de partida de una navegación física.

A medida que el usuario se mueva, en una navegación física, dentro del ambiente será sensado por otros objetos físicos; cuando un objeto físico sense un usuario (UserState *Navigating*), de acuerdo a la actividad (PhysicalActivity mencionada en 3.2) del usuario, el *PhysicalBrowserModel* actualizará su `currentNode` con el correspondiente *TravelNode*. Si el usuario se encuentra dentro del recorrido sugerido, entonces su PhysicalActivity será *InPath*, por lo tanto, se instanciará un nodo del tipo *IntermediateInPathTravelNode* o *IntermediateInPathPOTravelNode* dependiendo de si el objeto que sensó al usuario es un punto de interés (edificio, catedral, etc.) o un objeto físico puro (semáforo, cartel, etc.); en caso que el usuario se encuentre fuera del recorrido (*OutsidePath*), el nuevo `currentNode` será una instancia de *IntermediateOutsidePathTravelNode* o *IntermediateOutsidePathPOTravelNode*. La manera de finalizar una navegación es porque o bien el usuario llega a destino, o bien decide cancelar la navegación actual. En caso de arribar al objeto destino (target) de la navegación actual, el *PhysicalBrowserModel* actualizará su `currentNode` con un nuevo *FinishTravelNode*; en caso que el usuario decida cancelar la navegación, el nuevo `currentNode` de *PhysicalBrowserModel* será un *CancelTravelNode*.

En resumen, un Objeto Físico sensará al usuario y este podrá estar caminando o navegando, si se encuentra caminando, el *PhysicalBrowserModel* actualizará su `currentNode` con un nuevo *PhysicalObjectNode*; si se encuentra navegando lo hará con alguna subclase de *TravelNode*.

Se pensó en tener todos los nodos representados, ya que de esta manera se puede manejar todo a nivel de nodos. El interactuar con los distintos nodos le permite a la estrategia poder comportarse de manera distinta para cada nodo (mediante el método `interactWith`, como se mostrará más adelante). Si se tuviera un solo nodo general (*PhysicalNode*) la estrategia tendría que interactuar con la actividad del usuario para poder determinar que debe mostrar. Por esta razón se opto por tener la jerarquía de nodos, donde cada nodo sabe como mostrarse.

Dentro de un sistema de Hipermedia Física un usuario puede revisitar Objetos Físicos (por medio de Back y Next). Cuando el usuario se encuentre en una navegación de este tipo se le mostrará en pantalla (del dispositivo) la acción que

esté llevando a cabo (navegando por Back o Next). Por tal motivo, como se quería mostrar información específica se extendió la jerarquía de clase de *PhysicalNode* a partir de *TravelNode*. El resultado fue que por cada subclase de *TravelNode* se tendrán dos subclases nuevas que van a mostrar en pantalla la acción que el usuario esté llevando a cabo. Por ejemplo, en la Figura 32 se muestra la subclase *StartTravelNode* con sus dos subclases nuevas, una por cada posible acción posible. De esta manera, el usuario se podrá dar cuenta si, al iniciar una navegación física, lo está haciendo por primera vez, por un Back o por un Next. Ya que tendrá información especifica en cada nodo.

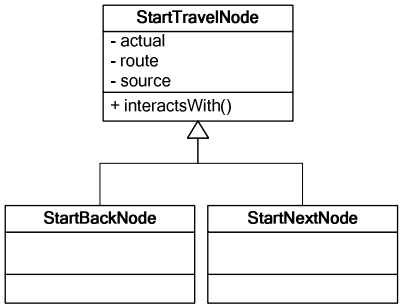


Figura 32: Jerarquía de clase de StartTravelNode.

Con esta jerarquía tenemos los nodos necesarios para visualizar cualquier acción posible de un usuario dentro de una navegación física (incluyendo Back y Next). Es importante aclarar estas nuevas subclases no redefinirán el método `interactsWith`, esto se debe a que adoptan el mismo comportamiento que su clase padre. Cada subclase que representa el Back y Next, agregan información en el html para ubicar al usuario en la tarea que esta desarrollando por ejemplo: "realizando un back".

4.2.3. Modelo de navegación físico.

Al igual que en 4.1.2 se definió el Modelo Físico *PhysicalBrowserModel*, en base al modelo *BasicBrowserModel* [Challiol et al, 2007] presentado en la sección 3.3. En la Figura 33 se muestra la incorporación del *PhysicalBrowserModel* sin tener en cuenta la parte digital presentada en 4.1.2 con el objetivo de focalizar la parte física independientemente de lo digital. Más adelante se integrarán los dos modelos en uno a fin de unificar conceptos (sección 4.3).

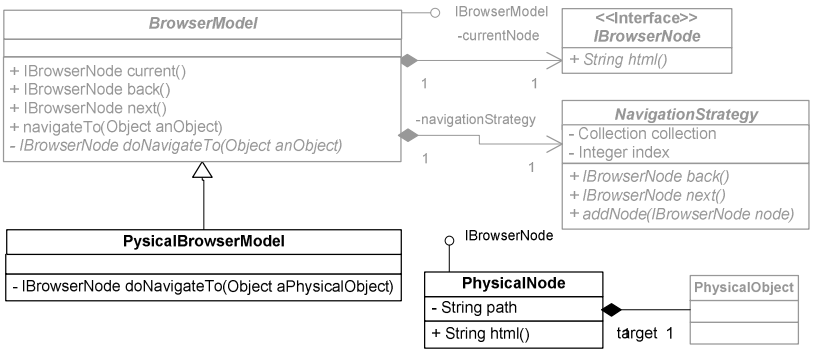


Figura 33: Physical Browser Model.

Como se ve en la Figura 33, en este nuevo modelo (al igual que en *DigitalBrowserModel*), se definieron dos nuevas clases, *PhysicalBrowserModel* y *PhysicalNode*. De la misma manera que un *DigitalNode* conoce su objeto digital, un *PhysicalNode* conoce su objeto físico actual y actúa como wrapper para tomar los datos necesarios del Objeto Físico (presentado en la sección 3.1) y mostrarlo, por ejemplo, en HTML.

La actualización del `currentNode` de un *PhysicalBrowserModel* se da cuando un usuario es sensado por un nuevo Objeto Físico. En la Figura 33 se ve que *PhysicalBrowserModel* es una subclase de *BrowserModel* y el `current` toma, en la instanciación de la clase *PhysicalNode*. Un desencadenante del sensado por parte de un nuevo Objeto Físico, es la actualización del `currentNode` de *DigitalBrowserModel*, siempre y cuando ese Objeto Físico sea un Punto de Interés y no un Objeto Físico Puro. En síntesis, cuando se actualice el `currentNode` de un *PhysicalBroserModel* por sensado de un POI se actualizará también el `currentNode` de *DigitalBrowserModel* con la contraparte digital correspondiente. Esta es otra manera de iniciar una navegación digital (como se menciono en 4.1.2) producto del sensado. Esta navegación es implícita ya que no involucra ninguna acción por parte del usuario.

En una navegación digital, los únicos métodos que actualizan el `currentNode` son `navigateTo`, `back` y `next;` e indirectamente, por el sensado por parte de un punto de interés. A diferencia de este tipo de navegación, dentro de una navegación física existen varios métodos que pueden actualizarlo. Esto se debe a que la actualización del `currentNode` puede provenir de distintas actividades físicas (actividades mencionadas en la sección 3.2).

Se enunciaran los desencadenantes de los métodos invocados por *PhysicalBrowserModel* que dan origen a la actualización de su `currentNode`. Estos métodos son producto de considerar el estado del usuario (Walking y Navigating) y su actividad dentro de este estado. La simulación del sensado hace que le llegue a la actividad del usuario (Walking y Navigating) el mensaje `updatePhysicalModel: aPhysicalModel with: aPhysicalObject`. A continuación se enunciaran que desencadena este mensaje.

Cabe aclarar que dentro del modelo de nodos físico existen dos clases especiales (junto a sus subclases) que representan los objetos físicos puros, ellas son *IntermediateInPathPOTravelNode* e *IntermediateOutsidePathPOTravelNode*. Ya que solo se consideraran importantes los objetos físicos puros dentro de una navegación física. Estos serán usados para informarle al usuario si se encuentra dentro o fuera de la navegación física elegida. Es decir, estos objetos físicos puros (PO) solo serán tenidos en cuenta por el modelo físico cuando un usuario se encuentre dentro de una navegación (*Navigating*); si el usuario se encuentra caminando (*Walking*) y un objeto físico puro lo sensa, el modelo físico no agregará dicho objeto a su historial de navegación.

Caso 1: Usuario esta caminando (*Walking*) y es sensado por un Punto de Interés.

A la actividad *Walking* le llega el mensaje, `updatePhysicalModel:`

`aPhysicalModel with: aPhysicalObject`, el Código 14 muestra este método. *Walking* tiene una sola actividad física posible que es *InFrontOf*, por eso delega en el modelo sin interactuar con su actividad (para determinar cual es).

```
updatePhysicalModel: aPhysicalModel with: aPhysicalObject
   aPhysicalModel navigateTo: aPhysicalObject.
```

Código 14: Método `updatePhysicalModel:aPhysicalBrowserModel with:aPhysicalObject` **de Walking.**

El Código 14 muestra como se invoca el método `navigateTo: aPhysicalObject` del modelo físico (*PhysicalBrowserModel*). Este método es el heredado de *BrowserModel.* Como se menciono en la sección 3.3, invoca al `doNavigateTo: aNodeDescription` (el cual debe ser definido por cada subclase *BrowserModel*). *PhysicalBrowserModel* define este método de la siguiente manera (ver Código 15).

```
doNavigateTo: aPhysicalObject
   ^PhysicalObjectNode for: aPhysicalObject path: self path.
```

Código 15: Método `doNavigateTo:aPhysicalObject` **de PhysicalBrowserModel.**

El Código 15, muestra que se crea un *PhysicalObjectNode* a partir del parametro `aPhysicalObject`. El nodo que devuelve es el que visualizara el Browser.

La Figura 34 muestra el diagrama de secuencia que se genera cuando el usuario es sensado por un punto de interés y el mismo se encuentra caminando.

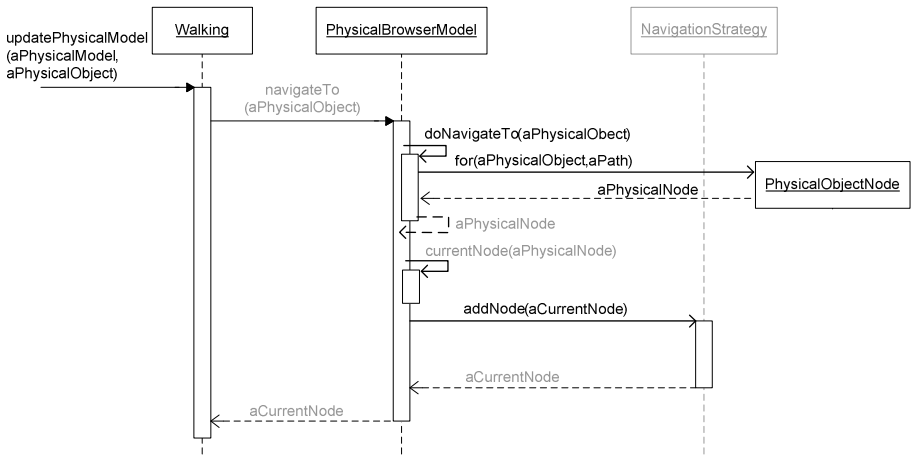


Figura 34: Diagrama de secuencia generado al ser sensado y su estado es Walking.

En la Figura 34 (en gris se hace referencia a la estructura heredada), el mensaje `updatePhysicalModel: aPhysicalModel with: aPhysicalObject` es el mensaje que le llega a *Walking* cuando el usuario está caminando y es sensado por un nuevo Punto de Interés (la simulación del sensado dispara este mensaje). Se puede ver en la Figura 34 que se sigue la secuencia de mensajes de acuerdo al Código 14. En la Figura 34 todavía no se hace hincapié en la política de navegación física, eso será presentado en la sección 4.2.4.

Caso 2: Usuario navegando un link físico (*Navigating*) y es sensado por un Punto de Interés.

Ahora veamos el caso de que el usuario se encuentre navegando (*Navigating*, producto de haber seleccionado un link físico). *Navigating* delega la resolución de `updatePhysicalModel: aPhysicalModel with: aPhysicalObject` en su actividad la cual decidirá que método debe de invocar del modelo. Esto puede visualizarse en el Código 16.

```
updatePhysicalModel: aPhysicalModel with: aPhysicalObject
  self activity updatePhysicalModel: aPhysicalModel with:
  aPhysicalObject
```

Código 16: Método `updatePhysicalModel:aPhysicalBrowserModel with:aPhysicalObject` **de Navigating.**

Las actividades posibles en las cuales delega el Código 16, son: *InPath*, *OutsidePath, ReachTarget.* Ya que una vez que el usuario inició una nueva navegación física, puede ir moviéndose por el ambiente en busca del `target` (basándose en el recorrido sugerido). Estos movimientos harán que los Objetos Físicos lo sensen. A continuación se mostrara como define cada una de estas actividades físicas el método `updatePhysicalModel: aPhysicalModel with: aPhysicalObject.`

Caso 2.1: Usuario navegando un link físico (*Navigating*) y es sensado por un Punto de Interés que esta dentro del camino (entonces se tendrá como actividad física: *InPath*).

InPath define el método `updatePhysicalModel: aPhysicalModel with: aPhysicalObject` como se muestra en el Código 17.

```
updatePhysicalModel:aPhysicalModel with:aPhysicalObject
  aPhysicalModel
  updateNavigateToWhenTravelInPath:aPhysicalObject.
```

Código 17: Método `updatePhysicalModel:aPhysicalBrowserModel with:aPhysicalObject` **de InPath.**

El Código 17 invoca el método `updateNavigateToWhenTravelInPath: aPhysicalObject` de *PhysicalBrowserModel.* La definición de este método se puede visualizar en el Código 18.

```
updateNavigateToWhenTravelInPath: aPhysicalObject
   self currentNode: (self currentNode
   createIntermedianteInPathNode:aPhysicalObject).
   self navigationModel addNode: self currentNode.
   self triggerEvent: #inTravel .
   ^self currentNode.
```

Código 18: Método `updateNavigateToWhenTravelInPath:aPhysicalObject` **de PhysicalBrowserModel.**

En el Código 18 se puede ver como se actualiza el `currentNode` con un nuevo nodo, el mismo es creado enviándole al `currentNode` el mensaje `createIntermedianteInPathNode:aPhysicalObject`, ver Código 19. Luego delega, en su modelo de navegación, el insertado del `currentNode`. Una vez actualizado el historial de navegación, se dispara el trigger `#inTravel` el cuál será interceptado por la Ventana que representa la parte física (*PhysicalWebView*) con motivo de actualizar sus botones de navegación correspondientes.

```
createIntermediateInPathNode: aPhysicalObject
   ^(IntermediateInPathTravelNode for:self
   actual:aPhysicalObject).
```

Código 19: Método `createIntermediateInPathNode:aPhysicalObject` de IntermediateTravelNode.

En el Código 19 se puede ver el método utilizado por un *TravelNode* para crear un *IntermediateInPathTravelNode* usando `aPhysicalObject` y él mismo (para que el nodo que se generare pueda tomar los datos que necesite del nodo actual al cual le llego el mensaje).

La Figura 35 muestra el diagrama de secuencia que se genera cuando el usuario es sensado por un punto de interés y el mismo se encuentra en el camino correcto dentro de una navegación:

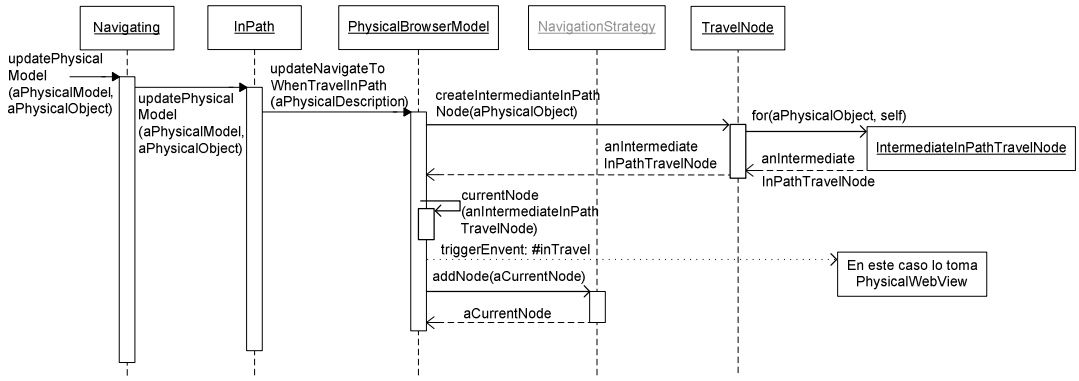


Figura 35: Diagrama de secuencia generado al ser sensado por un PO y estado Navigating.

En la Figura 35, el mensaje `updatePhysicalModel: aPhysicalModel with: aPhysicalObject` es el mensaje que le llega a *InPath* (*Navigating* delega en su actividad actual) cuando el usuario se encuentra en una navegación y es sensado por un nuevo Punto de Interés que se encuentra dentro de la ruta sugerida. Se puede ver en la Figura 35 que se sigue la secuencia de mensajes de acuerdo al Código 17, 18 y 19. En la Figura 35 todavía no se hace hincapié en la política de navegación física, eso será presentado en la sección 4.2.4.

En este caso, el usuario es sensado por un Punto de Interés. Si hubiera sido sensado por un Objeto Físico puro, la secuencia de mensajes habría sido muy similar (diferenciándose en los nombres que identificarían la interacción con un

Objeto Físico puro) y cambiaria particularmente en la creación de un *IntermediateInPathPOTravelNode* en vez de un *IntermediateInPathTravelNode.*

Caso 2.2: Usuario navegando un link físico (*Navigating*), y es sensado por un Punto de Interés que esta fuera del camino (entonces se tendrá como actividad física: *OutsidePath*).

OutsidePath define el método `updatePhysicalModel: aPhysicalModel with: aPhysicalObject` como se muestra en el Código 20.

```
updatePhysicalModel:aPhysicalModel with:aPhysicalObject
   aPhysicalModel updateNavigateToWhenTravelOutsidePath:
   aPhysicalObject.
```

Código 20: Método `updatePhysicalModel:aPhysicalBrowserModel with:aPhysicalObject` **de OutsidePath.**

El Código 20 invoca el método `updateNavigateToWhenTravelOutsidePath: aPhysicalObject` de *PhysicalBrowserModel.* La definición de este método se puede visualizar en el Código 21.

```
updateNavigateToWhenTravelOutsidePath: aPhysicalObject
   self currentNode: (self currentNode
   createIntermedianteOutsidePathNode:aPhysicalObject).
   self navigationModel addNode: self currentNode.
   self triggerEvent: #inTravel .
   ^self currentNode.
```

Código 21: Método `updateNavigateToWhenTravelOutsidePath:PhysicalObject` **de PhysicalBrowserModel.**

En el Código 21 se ve como se actualiza el currentNode mediante el mensaje `createIntermedianteOutsidePathNode(aPhysicalObject)` que se envía al nodo actual (*TravelNode*). Una vez creado el nuevo nodo delega, en su modelo de navegación, el insertado del currentNode. Al igual que en Código 18, una vez actualizado el historial de navegación, se dispara el trigger `#inTravel` el cuál será interceptado por la Ventana que representa la parte física (PhysicalWebView) con motivo de actualizar sus botones de navegación correspondientes.

En el Código 22 se puede ver el método utilizado por un *TravelNode* para crear un `IntermediateOutsidePathTravelNode` usando `aPhysicalObject` y él mismo (para que el nodo que se generare pueda tomar los datos que necesite del nodo actual al cual le llego el mensaje).

```
createIntermediateOutsidePathNode: aPhysicalObject
   ^(IntermediateOutsidePathTravelNode for:self
   actual:aPhysicalObject).
```

Código 22: Método `createIntermediateInPathNode:aPhysicalObject` **de IntermediateTravelNode.**

La secuencia de mensajes (diagrama de secuencia) es muy parecida a la presentada para el caso 2.1, por tal motivo no se agrego un nuevo diagrama.

En este caso, el usuario es sensado por un Punto de Interés. Si hubiera sido sensado por un Objeto Físico puro, la secuencia de mensajes habría sido muy similar (diferenciándose en los nombres que identificarían la interacción con un Objeto Físico puro) y cambiaria particularmente en la creación de un *IntermediateOutsidePathPOTravelNode* en vez de un *IntermediateOutsidePathTravelNode.*

Caso 2.3: Usuario navegando un link físico (*Navigating*) y es sensado por un punto de interés que es el target buscado (entonces se tendrá como actividad física: *ReachTarget*).

ReachTarget define el método `updatePhysicalModel: aPhysicalModel with: aPhysicalObject` como se muestra en el Código 23.

```
updatePhysicalModel:aPhysicalModel with: aPhysicalObject
   aPhysicalModel navigateToWhenFinishedTravel:
   aPhysicalObject.
```

Código 23: Método `updatePhysicalModel:aPhysicalBrowserModel with:aPhysicalObject` **de ReachTarget.**

El Código 23 invoca el método `navigateToWhenFinishedTravel: aPhysicalObject` de *PhysicalBrowserModel.* La definición de este método se puede visualizar en el Código 24.

```
navigateToWhenFinishedTravel: aPhysicalObject
   self currentNode:(self currentNode
   createFinishNode:aPhysicalObject ).
   self navigationModel addNode: self currentNode.
   self triggerEvent: #finishedTravel .
   ^self currentNode.
```

Código 24: Método `navigateToWhenFinishedTravel:PhysicalObject` **de PhysicalBrowserModel**

El Código 24 actualiza su `currentNode` con un nuevo nodo *FinishTravelNode*, este nodo es creado en `createFinishNode:aPhysicalObject` como se muestra en el código 25. Luego delega, en su modelo de navegación, el insertado del `currentNode`. Una vez actualizado el historial de navegación, se dispara un trigger `# finishedTravel`, el mismo será interceptado el cuál será interceptado por la Ventana que representa la parte física (*PhysicalWebView*).

```
createFinishNode: aPhysicalObject
   ^(FinishTravelNode for:self actual:aPhysicalObject).
```

Código 25: Método `createFinishNode:aPhysicalObject` **de IntermediateTravelNode.**

En el Código 25 se puede ver el método utilizado por un *TravelNode* para crear un *FinishTravelNode* usando `aPhysicalObject` y él mismo (para que el nodo que se generare pueda tomar los datos que necesite del nodo actual al cual le llego el mensaje).

No se agrego un nuevo diagrama ya que la secuencia de mensajes (diagrama de secuencia) es muy parecida a la presentada para el caso 2.1.

Caso 3: Usuario esta caminando (*Walking*) y clickea un link físico.

En el caso de que el usuario se encuentre caminando y decida navegar explícitamente a un punto de interés por medio de clickear en link físico se dará origen a una nueva navegación física por medio de `navigateToWhenTravel: aTravel` (ver Código 26) que se le envía al *PhysicalBrowserModel*. Este mensaje es invocado desde la ventana cuando se clickea el link.

```
navigateToWhenTravel: aTravel
   self currentNode:(StartTravelNode for: aTravel path: self
   path).
   self navigationModel addNode: self currentNode.
   ^self currentNode.
```

Código 26: Método `navigateToWhenTravel: aTravel` de PhysicalBrowserModel.

El Código 26 actualiza su `currentNode` con un nuevo nodo *StartTravelNode*, este nodo es creado pasándole `aTravel` (representa un objeto que conoce origen, destino y objetos intermedios del camino) que llego como parámetro y el path donde se encuentran los templates. Luego delega, en su modelo de navegación, el insertado del currentNode.

En la Figura 36, el mensaje `processLink:anURL` es el mensaje que se invoca en la vista física *PhyhicalWebView* cuando el usuario hace click en un link físico. Se puede ver en la Figura 36 que se sigue la secuencia de mensajes, a partir de *PhysicalBroswerModel*, de acuerdo al Código 26. También se puede visualizar como reacciona la ventana (*PhysicalWebView*) cuando le llega el mensaje `processLink:anURL`, este mensaje es capturado antes de que se ejecute para actualizar la actividad del usuario (pasando a estar *Navigating*). La ventana interactua con la actividad del usuario para poder obtener el travel generado (el cual sabe el origen, destino y objetos intermedios del camino). En cuanto al mensaje `startTravel` actualiza los estados de los botones.

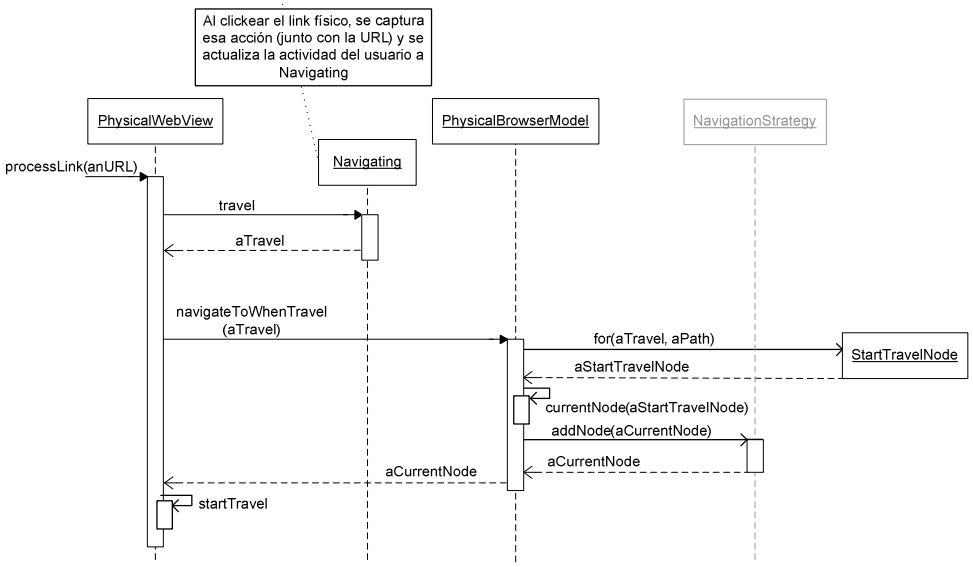


Figura 36: Diagrama de secuencia generado al clickear un link físico (Walking).

Caso 4: Usuario navegando un link físico (*Navigating*), y cancela la navegación.

El usuario, estando dentro de una navegación física, puede cancelarla en cualquier momento, en ese caso se invoca `cancelNavigation: aCurrentNode` (ver Código 27) del *PhysicalBrowserModel* desde la ventana (*PhysicalWebView*).

```
cancelNavigation: aCurrentNode
   self currentNode: (self currentNode createCancelNode:
   aCurrentNode).
   self navigationModel addNode: self currentNode
```

Código 27: Método `cancelNavigation:PhysicalNode` de PhysicalBrowserModel.

El Código 27 actualiza su currentNode con un nuevo nodo *CancelTravelNode*, creado mediante el mensaje `createCancelNode(aCurrentNode)` (ver código 28) que llega al *TravelNode* actual. Luego delega, en su modelo de navegación, el insertado del `currentNode`.

```
createCancelNode: aCurrentNode
   ^(CancelTravelNode for: aCurrentNode path: self path).
```

Código 28: Método `cancelNavigation:aCurrentNode` de IntermediateTravelNode.

En el Código 28, es el mensaje que le llega a un *TravelNode* para que crea un `CancelTravelNode` usando `aCurrentNode` y el path donde se encuentran los templates.

Se puede ver en la Figura 37 que se sigue la secuencia de mensajes, a partir de *PhysicalBroswerModel*, de acuerdo al Código 27. También se puede visualizar como reacciona la ventana (*PhysicalWebView*) cuando le llega el mensaje `cancelPH`, este mensaje es capturado antes de que se ejecute para actualizar la actividad del usuario (pasando a estar Walking). En cuanto al mensaje `finishTravel` actualiza los estados de los botones.

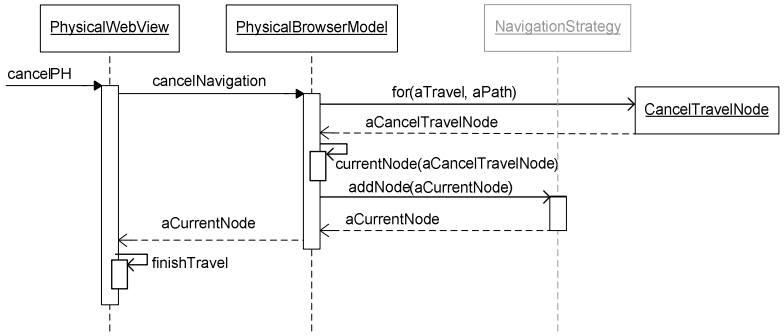


Figura 37: Diagrama de secuencia generado al cancelar una navegación física.

Caso 5: Usuario navegando un link físico (*Navigating*) y hace back en la navegación.

El usuario, estando dentro de una navegación física, puede hacer back en cualquier momento, en ese caso se invoca `back` del *PhysicalBrowserModel* desde la ventana (*PhysicalWebView*).

```
back
   | aTargetNode |
   aTargetNode := self navigationModel back.
   self currentNodeTemporal: StartBackNode forOrigin:  self
   CurrentNode target: aTargetNode.
   self navigationModel addNodeWithoutUpdateNavigationIndex:
   self currentNode.
   ^self currentNode actualPhysicalObject.
```

Código 29: Método `back` de PhysicalBrowserModel.

Lo primero que hace el código 29 es recuperar el nodo anterior; esto lo hace por medio del método `back` del modelo de navegación. Éste nodo anterior tendrá el nuevo punto de interés destino a visitar y será utilizado para crear un nuevo *StartBackNode* (a partir del nodo actual y al que se quiere volver); una vez que dicho nodo fue creado actualiza su `currentNode`. Posterior a esto delega, en su modelo de navegación, el insertado del nuevo `currentNode` al historial de navegación (ya que no se hizo un navegación real usa el mensaje `addNodeWithoutUpdateNavigationIndex` para que no se actualicen los índices, simplemente agrega el nodo).

En la Figura 38, el mensaje `goBackPH` es el mensaje generado por la vista física *PhyhicalWebView* cuando el usuario decide hacer back. Se puede ver en la Figura 38 que se sigue la secuencia de mensajes, a partir de *PhysicalBroswerModel*, de acuerdo al Código 29. También se puede visualizar como reacciona la ventana (*PhysicalWebView*) cuando le llega el mensaje `backPH`, este mensaje es capturado antes de que se ejecute para actualizar la actividad del usuario. En cuanto al mensaje `startTravel` actualiza los estados de los botones.

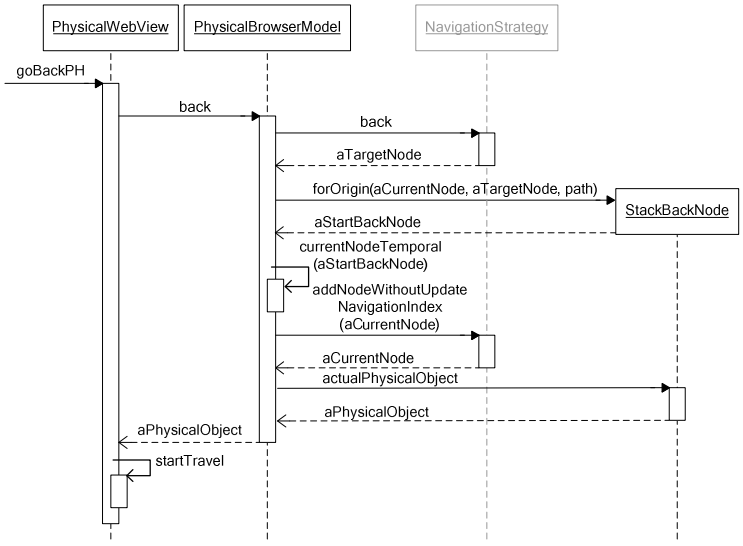


Figura 38: Diagrama de secuencia generado al hacer back.

Caso 6: Usuario navegando un link físico (*Navigating*) y hace next en la navegación.

El usuario, estando dentro de una navegación física, puede hacer next en cualquier momento, en ese caso se invoca `next` (ver Código 30).

```
next
   | targetNode |
   targetNode := self navigationModel next.
   self currentNodeTemporal: StartNextNode forOrigin:  self
   CurrentNode target: aTargetNode.
   self navigationModel addNodeWithoutUpdateNavigationIndex:
   self currentNode.
   ^self currentNode actualPhysicalObject.
```

Código 30: Método `next` de PhysicalBrowserModel.

El código 30 es análogo al código 29 a excepción que, en vez de buscar el nodo anterior, busca el siguiente y en vez de crear un nodo *StartBackNode*; crea un *StartNextNode*. En Código 30 actualiza su `currentNode` con un nuevo nodo *StartNextNode*.

El usuario debe de realizar al menos un Back (seleccionar el botón) para poder tener habilitado el botón de Next. El Next avanza sobre el retroceso, permite elegir el nodo al que se quiere volver. Es decir, tanto el Back como el Next siempre el usuario va a estar volviendo a nodos visitados.

Tanto para el caso 5, como 6 inician una nueva navegación (de Back o Next) por lo tanto realizando esta actividad pueden darse los mismo casos planteados para la navegación de un link físico (caso 2). La diferencia estará en la secuencia de mensajes (diferenciándose en los nombres usados para la interacción) y cambiara particularmente la creación de los distintos nodos asociados al Back y Next.

Hasta el momento se presentaron distintos casos para que el *PhysicalBrowserModel* actualize el `currentNode`, en todos estos delega en la estrategia de navegación, el insertado del `currentNode` en la lista de Nodos visitados. En la sección 4.2.4, veremos como se implementa las estrategias o políticas de navegación física.

4.2.4. Política de navegación física (*ConfigPhysicalStrategy*).

En [Challiol et al, 2007] se mencionan estrategias de navegación física aplicables al momento de revisitar un punto de Interés; se pueden encontrar, entre otras, dos estrategias: una primer estrategia que guardaría todos los objetos visitados por el usuario (*FullPhysicalStrategy*) y una segunda que solo guardaría los objetos de interés (*InterestPhysicalStrategy*), de acuerdo a algún tipo de interés (cultural, arquitectónico, deportivo, etc.), previamente definidos por él. La situación se podría volver mas complicada si se quisieran filtrar objetos acordes a información del contexto (como ser la actividad o estados del usuario, enunciada en 3.2). Esto daría como resultado nuevas políticas de navegación las cuales solo almacenarían objetos acordes a la actividad del usuario; por ejemplo, se podría tener una nueva

política de navegación (*ReachTargetStrategy*) que solo almacene aquellos objetos físicos a los cuáles un usuario alcanzó (como target de una navegación). A simple vista se podrían tener varias políticas o estrategias de navegación diferentes.

Por otro lado, a lo largo de 4.1.3 se enunciaron y desarrollaron dos estrategias de navegación digital (Stack-Based y Receny-Based); las mismas quedaron representadas dentro del modelo *DigitalBrowserModel*. Como resultado, cada estrategia de navegación está representada por una subclase de *NavigationStrategy*.

Se podrían diseñar nuevas políticas o estrategias de navegación físicas como subclases de *NavigationStrategy*. Cada nueva política (clase) tendría su propia funcionalidad aunque habría demasiada similaridiidad entre ellas. Se analizó que solo se diferenciaban en el filtrado que se le aplicaba a los nodos visitados. Es decir, si fueron targets alcanzados, si son del grupo de interés del usuario o en el caso de ser la estrategia *FullPhysicalStrategy* considerar todos nodos. También se analizo poder cambiar de estrategia cuando el usuario esta usando la aplicación, entonces para esto se deben de tener todos los nodos para poder cambiar de estrategia en forma dinámica.

Como resultado se decidió tener una única estrategia de navegación, configurable a las necesidades del usuario. Que tenga todos los nodos visitados, y los filtre según la configuración del usuario (este filtrado se tiene en cuenta a la hora de hacer back y next)

Al igual que las políticas desarrolladas en 4.1.3, se tomó el diagrama presentado en Figura 15 (sección 3.3) y se creó la clase *ConfigPhysicalStrategy* (Como subclase de la clase *NavigationStrategy*) con el fin de agregar una política de navegación física al modelo existente. La Figura 39 muestra la nueva clase *ConfigPhysicalStrategy*.

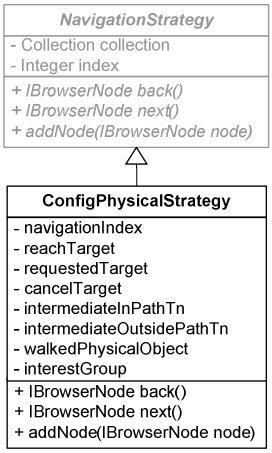


Figura 39: ConfigPhysicalStrategy en el Modelo.

A diferencia de las políticas presentadas en 4.1.3, la nueva estrategia de navegación, *ConfigPhysicalStrategy*, será configurable. El motivo de tal es brindar una política adaptable a las preferencias del usuario.

En primer instancia, la configurabilidad de la misma se puede subdividir en dos grupos: objetos de interés y opciones de navegabilidad.

Para el primer grupo, objetos de interés, el usuario seleccionará los grupos de objetos que él considere de su interés; un grupo de interés está representado por una colección de Objetos Físicos que comparten alguna característica en común; por ej: grupo de Objeto Cultural, Arquitectónico, Recreativo, Deportivo, etc. Supongamos que el usuario está interesado en recorrer objetos culturales, entonces seleccionará el grupo de interés Objeto Cultural. El usuario puede seleccionar más de un grupo, por ej: selecciona grupo de interés Objeto Cultural y Deportivo. Estos grupos de interés se guardan en `interestGroup`. Entonces cuando el usuario hace back y next solo se consideran nodos visitados incluidos en `interestGroup`.

El segundo grupo configurable (opciones de navegabilidad) son un conjunto de variables (booleanas) que el usuario setea según sus preferencias. Las variables son:

- `reachTarget`: Representa si se deben considerar los nodos que fueron creados al alcanzar un target deseado.
- `requestedTarget`: Representa si se deben considerar los nodos que fueron creados al cliquear un link físico.
- `cancelTarget`: Representa si se deben considerar los nodos que fueron creados al cancelar una navegación física.
- `intermediateInPathTn`: Representa si se deben considerar los nodos que estaban en el camino del usuario cuando realizo una navegación física.
- `intermediateOutsidePathTn`: Representa si se deben considerar los nodos que estaban fuera del camino del usuario cuando una navegación física.
- `walkedPhysicalObject`: Representa si se deben considerar los nodos que fueron creados porque el usuario estaba caminando y fue sensado por un punto de interés.

Cada una de estas opciones representa el interés del usuario sobre los nodos. Por ejemplo, si el usuario seteó la opción `walkedPhysicalObject` indica que le interesa, ante un posible Back o Next, tener en cuenta a los nodos de este tipo (*PhysicalObjectNode*); si también seteó la opción `reachTarget` indica que le interesa tener en cuenta a los nodos de este tipo (*FinisihTravelNode*); si setea `intermediateInPathTn` indica que le interesa tener en cuenta a aquellos nodos en los cuáles estuvo en camino correcto dentro de alguna navegación (*IntermediateInPathTravelNode*), etc. A los nodos visitados que el usuario decida no tener en cuenta simplemente no seteará su opción. A manera de ilustrar esto con un ejemplo extremo, supongamos un usuario que no seleccionó ninguna de las opciones de navegabilidad, por lo tanto nunca existirá la posibilidad de revisitar un Nodo; gráficamente esto se verá en la inhabilitación constante de los botones de Back y Next.

Estas configuraciones son tenidas en cuenta cuando un usuario selecciona Back o Next, momento en el cuál se debe buscar, por medio de la política

ConfigPhysicalStrategy, el objeto adecuado según la configuración, el cual le será mostrado.

A continuación se muestra un caso particular de configuración, utilizando los ejemplos expuestos en 4.2.1., a fin de mostrar como, por medio de las diferentes configuraciones de la política, se pueden obtener distintos resultados, en cuanto a objetos físicos a revisitar. La Figura 40 muestra el historial de navegación actual de la política.

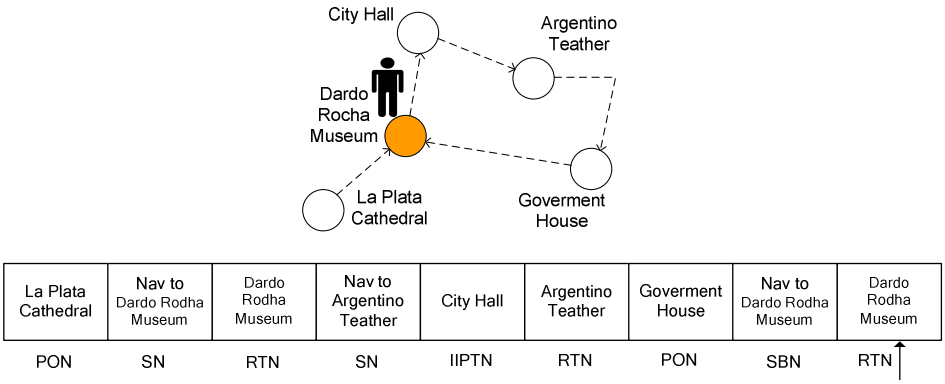


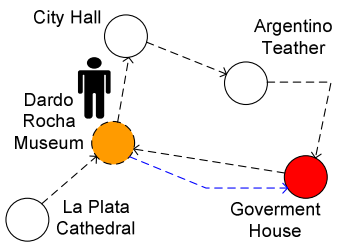
Figura 40: Historial de una navegación física.

En Figura 40 se puede ver, debajo de los nodos, la clase de nodo a la que pertenece; en este ejemplo hay *PhysicalObjectNode* (PON), *ReachTargetNode* (RTN), *IntermediateInPathTravelNode* (IIPTN), *StartTravelNode* (SN) y *StartBackNode* (SBN). A manera de simplificar el ejemplo supongamos que la política no tiene asociado ningún grupo de interés particular.

Ejemplo 1: Supongamos que el usuario seteó las siguientes opciones de navegabilidad:

- ✓ `reachTarget`
- x `requestedTarget`
- x `cancelTarget`
- x `intermediateInPathTn`
- x `intermediateOutsidePathTn`
- ✓ `walkedPhysicalObject`

El usuario decide hacer un back, por lo tanto, en base a la configuración de su política, el nodo a revisitar será 'Goverment House' (fue alcanzado caminando).



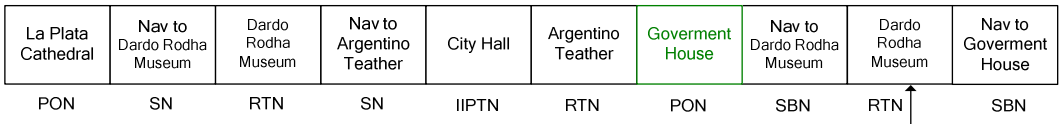


Figura 41: Historial de una navegación física (primer back).

Nuevamente, el usuario decide hacer otra vez back, por lo tanto el nuevo objeto a revisitar, en base a su política, será 'Argentino Teather' (target alcanzado); esto se ve en Figura 42.

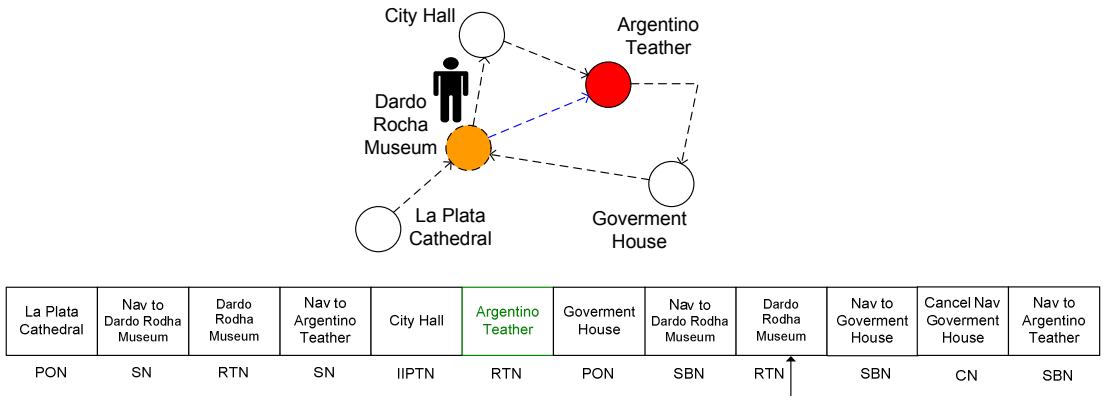
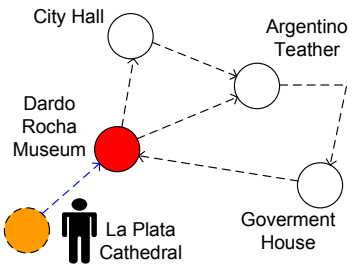


Figura 42: Historial de una navegación física (segundo back)

En la Figura 42, se puede apreciar que el puntero indica donde estoy parado, y se agregaron dos navegaciones producidas por un back (una de ellas cancelada, agregándose el nodo CN-CancelNavigation), la navegación que queda vigente es la ultima activa las anteriores no tendrán valides (es decir, se cancelan).

Según muestra la Figura 42, después de hacer dos veces back, el usuario se encuentra en una nueva navegación cuyo target es 'Argentino Teather' (RTN). Una vez iniciada la nueva navegación el usuario emprende su viaje en busca de 'Argentino Teather' pero, por descuido, camina en dirección contraria y llega a 'La Plata Cathedral' (IOPTN), en ese momento ve en pantalla que se encuentra fuera del recorrido con lo cuál, contrariado, decide hacer un Back. Al seleccionar el nuevo Back se cancela la navegación actual y se inicia una nueva a partir de la posición actual (La Plata Cathedral). Este Back referencia al 'Dardo Rocha Museum' ya que es el último punto de interés visitado y que cumple con la configuración (ya que fue alcanzado porque era target de un link físico). En la Figura 43 se puede visualizar esta situación y como quedo el historial de navegación.



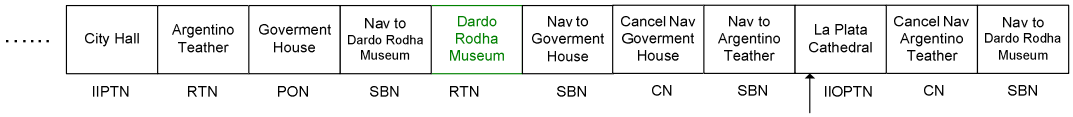


Figura 43: Historial de una navegación física (primer back desde 'La Plata Cathedral')

En Figura 43 se cancela la navegación actual y luego, en base a la configuración de la política, se da inicio a una nueva navegación que va desde 'La Plata Cathedral' (`source`) a 'Dardo Rocha Museum' (`target`). Llegando finalemente a 'Dardo Rocha Museum'.

Ejemplo 2: Supongamos cambia dinámicamente la política seteando las siguientes opciones de navegabilidad:

- x `reachTarget`
- x `requestedTarget`
- x `cancelTarget`
- ✓ `intermediateInPathTn`
- x `intermediateOutsidePathTn`
- x `walkedPhysicalObject`

El usuario decide hacer un Back, por lo tanto, en base a la configuración de su política, el nodo a revisitar será 'City Hall' (ya que es un IIPTN); esto se ve en Figura 44.

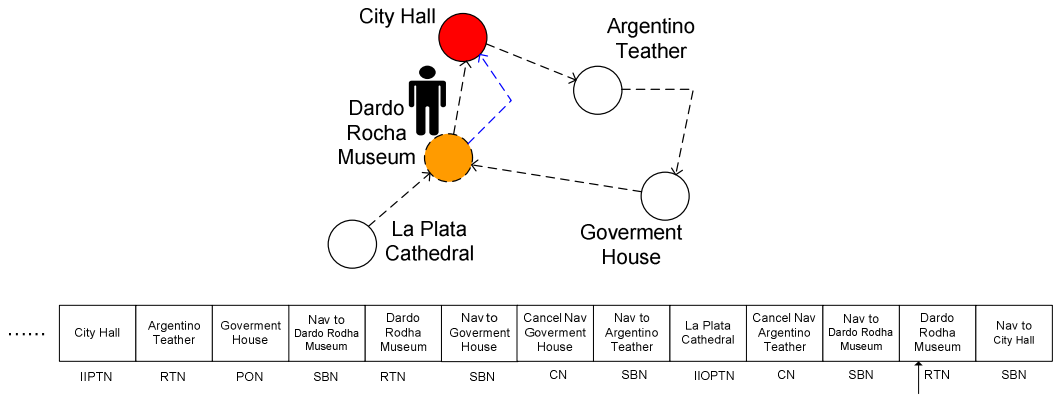


Figura 44: Historial de una navegación física.

Como se puede ver en el historial de navegación, en Figura 44, solo hay un nodo IIPTN, por lo tanto, una vez que el usuario hizo un click en back, se deshabilitan los botones de back y Next debido a que no existen otros posibles nodos a revisitar en base a la configuración de la política de este ejemplo.

Como conclusión, en estos dos ejemplos se vió como, en base a un único historial de navegación, existen diferentes posibilidades de revisitar un nodo dependiendo de la configuración de la política de navegación.

A continuación se muestran los métodos de navegación que debe definir la política *ConfigPhysicalStrategy*:

```
addNode: aBrowserNode
   self collection add: aBrowserNode.
   self index: self index + 1.
   self navigationIndex: self index.
   ^self currentNode.
```

Código 31: Método `addNode:IBrowserNode` **de ConfigPhysicalStrategy.**

El Código 31 muestra como un nodo físico es agregado al historial de navegación de la política. Es invocado por *PhysicalBrowserModel* cuando detecta que fue sensado por un nuevo punto de interés independientemente de la actividad del usuario. Agrega todos los nodos visitados, el `back` y `next` se encargaran de filtrar según la configuración.

El Código 32 (`back`) implementa un retroceso en el historial de navegación retornando un nuevo nodo actual, `node`; éste se buscará dentro de la collección `aux` integrada por todos los posibles nodos físicos a revisitar; `aux` incluye todos los nodos físicos anteriores a la posición de `navigationIndex`. Una vez armada la colección se buscará, dentro de ésta el nodo a revisitar, por medio del método `isCurrentNode: aNode withActualPO: actualPO`. Este método busca considerando la configuración actual. Una vez detectado el nodo, se actualiza el índice de navegación `navigationIndex`.

```
back
   | aux node x |
   aux := (self collection copyFrom: 1 to: (self
   navigationIndex-1)) reverse.
   x := 0.
   node := aux detect:[:aNode | x:=x+1. ( self
   isCurrentNode:aNode withActualPO:(self currentNode
   physicalNode target) ) ] .
   self navigationIndex: self navigationIndex - x .
   ^node.
```

Código 32: Método `back` **de ConfigPhysicalStrategy.**

De manera análoga, el Código 33 implementa un avance en el historial de navegación retornando el nuevo nodo actual, `node`; La principal diferencia con el Código 32 se encuentra en el armado de la collección `aux;` se incluyen todos los nodos físicos posteriores a `navigationIndex`. También se llama al método `isCurrentNode: aNode withActualPO: actualPO` para considerar la configuración.

```
next
   | aux node x |
   aux := (self collection copyFrom: (self navigationIndex+1)
   to: (self collection size)) .
   x := 0.
   node := aux detect:[:aNode | x:=x+1. (self
   isCurrentNode:aNode withActualPO:(self currentNode
   physicalNode target) ) ] .
   self navigationIndex: self navigationIndex + x .
   ^node.
```

Código 33: Método `next` **de ConfigPhysicalStrategy.**

Los Códigos 31, 32 y 33 forman parte de los métodos de navegación de ConfigPhysicalStrategy. A continuación se mostrarán los principales métodos privados de esta política:

```
isCurrentNode: aNode withActualPO: actualPO
   ^ ( ( aNode actualPhysicalObject name = actualPO name) not &
   ( self includeNode: aNode ) & ( self inGroupObject: aNode ) )
```

Código 34: Método `isCurrentNode:aPhysicalNode withActualPO:aPhysicalObject` **de ConfigPhysicalStrategy.**

El Codigo 34 retorna un valor booleano indicando si `aNode` es un posible nodo a revisitar; para ello `aNode` debe ser distinto del nodo actual, tener seteada su clase dentro de las opciones de navegabilidad (mediante `includeNode: aNode`) y pertenecer al grupo de objetos de interés del usuario (`inGroupObject: aNode`). De manera conceptual, el código 34 define si `aNode` es tenido en cuenta por el usuario como posible Nodo a revisitar.

Utilizando las opciones de navegabilidad configuradas por el usuario, el próximo Código 35 retorna un valor booleano en base al nodo `aNode`. Este mecanismo implementa un double dispatch para decidir si será tenido en cuenta al momento de buscar el siguiente nodo.

```
includeNode: aNode
   ^aNode interactsWith: self.
```

Código 35: Método `includeNode:PhysicalNode` **de ConfigPhysicalStrategy.**

Veamos como se realiza el doble dispach generado a partir de `interactsWith: aNavigationModel` (que se dispara del Código 35). Por cuestiones de simplicidad solo se presentarán como éste método es definido por tres nodos distintos.

PhysicalObjectNode define el método `interactsWith:aNavigationModel` como lo muestra el Código 36.

```
interactsWith: aNavigationModel
   ^aNavigationModel interactsWithPhysicalObjectNode: self.
```

Código 36: Método `interactsWith:aNavigationModel` **de PhysicalObjectNode.**

El Código 36 envía a la política de navegación el mensaje `interactsWithPhysicalObjectNode: aPhysicalObjectNode` (Este método esta definido *ConfigPhysicalNavigationModel* como muestra el Código 37).

```
interactsWithPhysicalObjectNode: aNode
   ^self walkedPhysicalObject.
```

Código 37: Método `interactsWithPhysicalObjectNode` **de ConfigPhysicalNavigation.**

El Código 37 muestra que cuando llega el mensaje `interactsWithPhysicalObjectNode: aPhysicalObjectNode` se

devuelve el valor seteado en la variable `walkedPhysicalObject` (este valor determinara si ese nodo es considerado o no por el usuario).

Veamos otro nodo (`IntermediateInPathTravelNode`) que define el método `interactsWith: aNavigationModel` como lo muestra el Código 38.

```
interactsWith: aNavigationModel
   ^aNavigationModel
   interactsWithIntermediateInPathTravelNode:self.
```

Código 38: Método `interactsWith:aNavigationModel` de IntermediateInPathTravelNode.

El Código 38 envía a la política de navegación el mensaje `interactsWithIntermediateInPathTravelNode: aPhysicalObjectNode` (*ConfigPhysicalNavigationModel* define este método como muestra el Código 39).

```
interactsWithIntermediateInPathTravelNode: aNode
   ^self intermediateInPathTn.
```

Código 39: Método `interactsWithIntermediateInPathTravelNode` de ConfigPhysicalNavigation.

El Código 39 muestra que cuando llega el mensaje `interactsWithIntermediateInPathTravelNode:aPhysicalObjectNode` se devuelve el valor seteado en la variable `intermediateInPathTn` (este valor determinara si ese nodo es considerado o no por el usuario).

FinishTravelNode define el método `interactsWith: aNavigationModel` como lo muestra el Código 40.

```
interactsWith: aNavigationModel
   ^aNavigationModel interactsWithFinishTravelNode: self.
```

Código 40: : Método `interactsWith:aNavigationModel` de FinishTravelNode.

El Código 40 envía, a la política de navegación, el mensaje `interactsWithFinishTravelNode: aPhysicalObjectNode` (Este método esta definido en el Código 41).

```
interactsWithFinishTravelNode: aNode
   ^self reachTarget.
```

Código 41: Método `interactsWithFinishTravelNode` de ConfigPhysicalNavigation.

El Código 41 muestra que cuando llega el mensaje `interactsWithFinishTravelNode: aPhysicalObjectNode` se devuelve el valor seteado en la variable `reachTarget` (este valor determinara si ese nodo es considerado o no por el usuario).

Otro método usado en el Código 34 es `inGroupObject: aPhysicalNode`. Este método se define en el Código 42 y retorna un valor booleano en base a si `aPhysicalNode` se encuentra en algún grupo de Objetos de interés definidos por

el usuario. Si no hay grupos devuelve `true` ya que se considera todos los nodos como validos.

```
inGroupObject: aPhysicalNode
   self interestGroup detect:[ :g | g includes:(aPhysicalNode
   actualPhysicalObject) ] ifNone:[^false].
   ^true.
```

Código 42: Método `isGroupObject:PhysicalNode` de ConfigPhysicalStrategy.

A continuación, la Figura 45 muestra el diagrama de secuencia de una nueva navegación física (acción de "clickear un Back físico"). Se toma de base el diagrama de la Figura 38 y se especifica el `back` definido por *ConfigNavigationModel*. En la Figura 38 se especificaba *NavigationStratey* pero no se consideraba ninguna política física en particular, la Figura 45 se focaliza en especificar este comportamiento ya que el resto del comportamiento fue definido por la Figura 38.

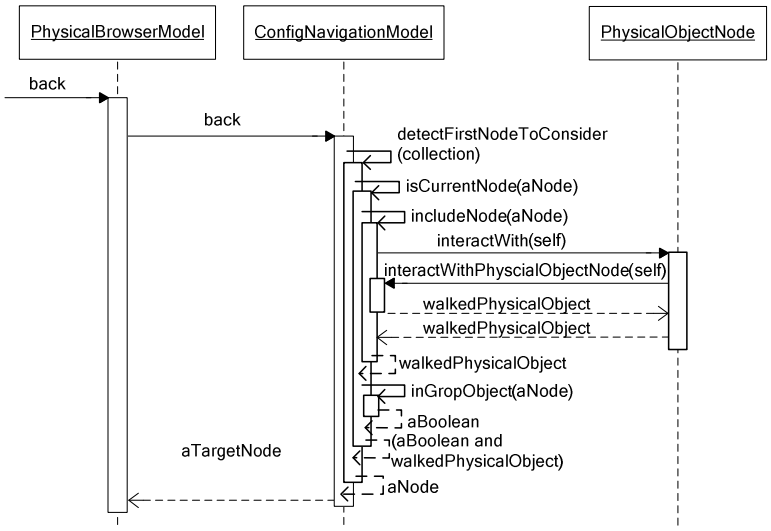


Figura 45: Diagrama de secuencia – Back Físico ConfigNavigationModel.

La Figura 45, muestra como es la interacción cuando se tiene como nodo a analizar un *PhysicalObjectNode* y la política consideraba estos como validos para realizar el regreso a los mismos.

4.3. Modelo para un Browser de Hipermedia Física.

A lo largo de este capítulo se fueron desarrollando modelos y políticas, tanto digitales como físicas, a fin de dar funcionalidad a un Browser de Hipermedia Física.

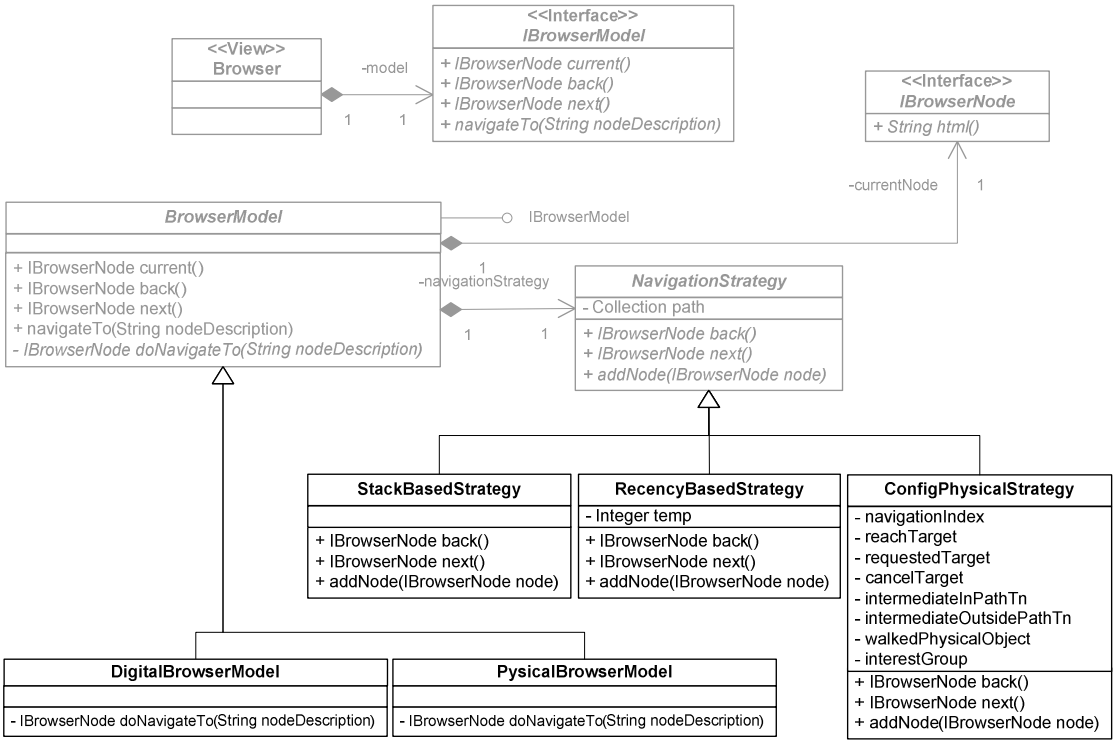
En primer lugar, partiendo del modelo *BasicBrowserModel* presentado en [Challiol et al, 2007] se desarrolló el modelo *DigitalBrowserModel* que, junto a las políticas digitales *StackBasedStrategy* y *RecencyBasedStrategy*, dieron funcionalidad al Browser en la parte de navegación digital.

Una vez resuelta la funcionalidad de un Browser digital se dió paso a la parte física. En primer lugar, se estudió en profundidad la semántica de una navegación física debido a la complejidad que ésta presenta respecto a la navegación digital. Una vez comprendida la semántica se diseño el *PhysicalBrowserModel*, un modelo para la navegación física.
Luego, en 4.2.3 se vió la necesidad de tener diferentes nodos físicos a fin de poder representar las diferentes actividades de un usuario. Por tal motivo se desarrolló una jerarquía de clases de nodos físicos. Esta jerarquía de clases sirvió para darle al usuario información detallada de lo que estaba haciendo.
Al final, se desarrolló la estrategia de navegación física *ConfigPhysicalStrategy* que junto al *PhysicalBrowserModel* dan funcionalidad al Browser en la parte de navegación física. Esta estrategia es configurable según los criterios más importantes del comportamiento de una aplicación de Hipermedia Física.

Como conclusión, se desarrolló un modelo que da funcionalidad al Browser de Hipermedia Física y se muestra a continuación, en la Figura 46.

Una importante ventaja de este modelo es la adaptabilidad del mismo. Supongamos que se quisiera agregar un nuevo modelo de navegación, para ello, bastaría con subclasificar *BrowserModel* y, de ser necesario, generar un nuevo nodo (o jerarquía de nodos) que cumpla con la interfaz *IBrowserNode*.

Por otro lado, supongamos que se quisiera diseñar una nueva política de navegación (digital, física u otra), habría que hacerlo subclasificando *NavigationStrategy*.



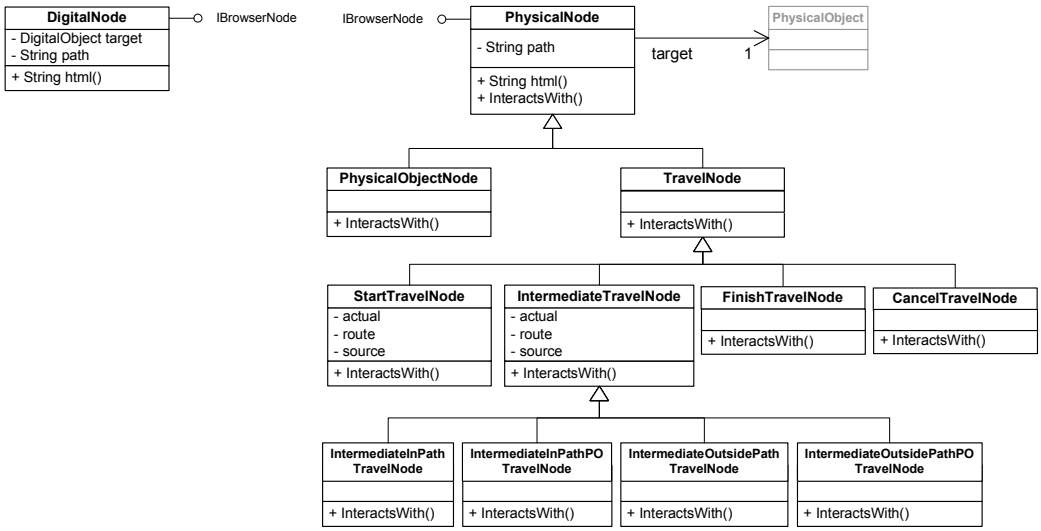


Figura 46: Modelo para Browser de Hipermedia Física.

Como se puede ver, el modelo presentado en esta tesis se adapta perfectamente a nuevas modificaciones o requerimientos sin modificar la implementación existente.

Capítulo 5. Ejemplo del Browser de Hipermedia Física.

5.1. Dominio de la Ciudad de la Plata

Para desarrollar el ejemplo se tomará como dominio el centro del casco urbano de la Ciudad de La Plata comprendido entre las Calles 6, 15, 49 y 54. La decisión de tales límites radica en la gran concentración de edificios turísticos y públicos dentro de este perímetro. Los puntos de interés a representar serán los siguientes: La Catedral, Museo de La Catedral, Museo Dardo Rocha, Torre 1, Torre 2, Palacio Municipal, Biblioteca Lopez Merino, Teatro Argentino, Legislatura, Tribunales, Casa de Gobierno y el Pasaje Dardo Rocha. Dentro del mapa se colocaron, a manera de ejemplo, tres semáforos (objetos físicos puros) ubicados en 12 esquina 53, 12 esquina 50 y 7 esquina 53. En Figura 47 se puede ver la vista inicial del Mapa.

En la Figura 47 se puede ver que el punto de interés “The Cathedral” está pintado de rojo, esto indica que el usuario fue sensado por este objeto. Es decir que el usuario se encuentra frente a “The Cathedral”, en ese momento el usuario puede ver una pantalla digital como se muestra en la Figura 48.

Figura 47: Vista inicial del Mapa de La Plata.

Figura 48: Vista digital y física de “The Cathedral”.

A manera de comprender mejor los ejemplos, se dividió a los mismos en dos secciones, en la primer sección se verán ejemplos de navegación digital y en la segunda todo lo relacionado a la parte física. En muchos casos, una actualización en la parte física implicará cambios en la parte digital.

5.2. Ejemplo Digital

Las operaciones posibles dentro de una navegación digital son clinkear en link digital, hacer back, next, reload o cancel. Como los requerimientos digitales

(clickear un link y recibir una respuesta) son manejados internamente y no requieren invocaciones HTTP, la respuesta que recibe el usuario esta totalmente controlada por el sistema de manera automática. Por tal motivo no se necesito darle un comportamiento particular al cancelar, ya que el mismo es casi imposible cliquearlo antes de recibir la respuesta (por la velocidad de la misma). En cuanto al reload muestra nuevamente el nodo actual.

Ahora se analizaran las posibles acciones del usuario y como el sistema reaccionara a las mismas.

5.2.1. El usuario clickea en link digital.

Estando ubicado en la catedral (“The Cathedral”) el usuario decide hacer click sobre el link digital "La Plata City", al hacer click sobre un link digital, se actualiza el `currentNode` del Browser. En la Figura 49 se ve la nueva página seleccionada por el usuario.

Estando en la página “La Plata City”, el usuario continuó navegando digitalmente por varios nodos, en la Figura 50 se muestra su historial de navegación digital.

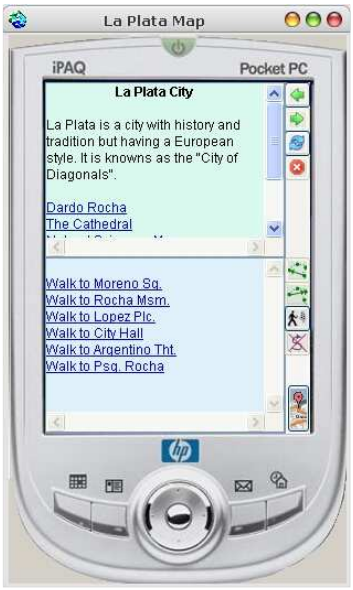


Figura 49: Vista de parte digital de la PDA.

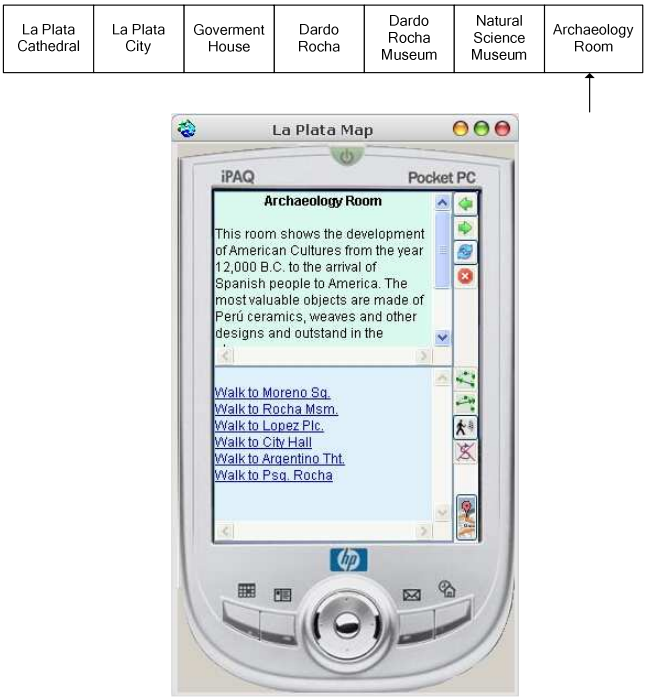


Figura 50: Historial de navegación digital de usuario.

5.2.2. El usuario hace back y next con política *StackBasedModel.*

A continuación se muestran distintos ejemplos dependiendo de la acción que tome el usuario usando como base el historial presentado en la Figura 50.

Caso 1: El usuario hace tres veces back digital.

Tomando como base el historial de navegación de la Figura 50; el nuevo *DigitalObject* a mostrar, después de hacer tres back, sería "Dardo Rocha" y el resultado de tal se ve en la Figura 51.

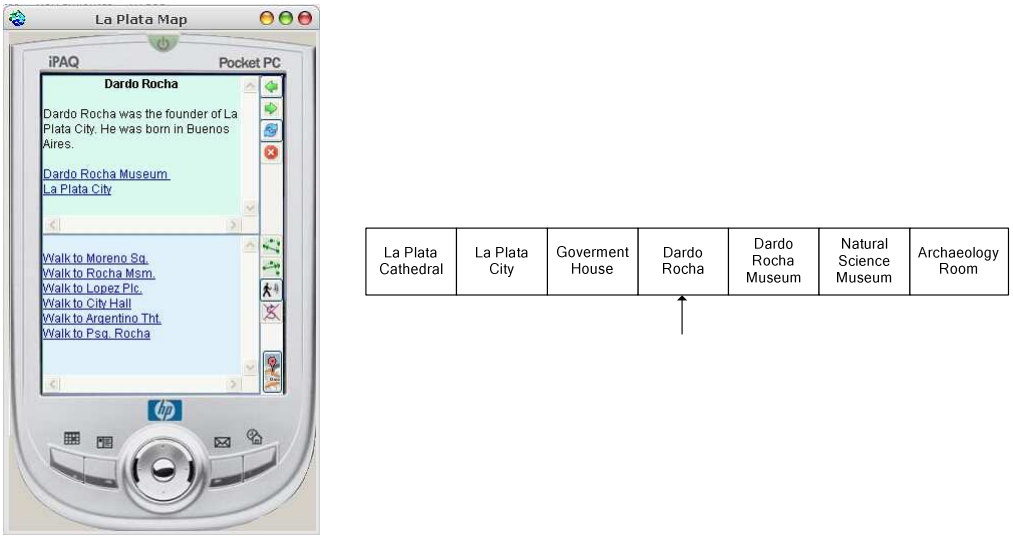


Figura 51: Vista digital después de hacer tres back.

Caso 2: El usuario hace un next digital.

Siguiendo con el ejemplo presentado en la Figura 51; el nuevo *DigitalObject* a mostrar, después de hacer un next, sería "Dardo Rocha Museum" y el resultado de tal se ve en la Figura 52.

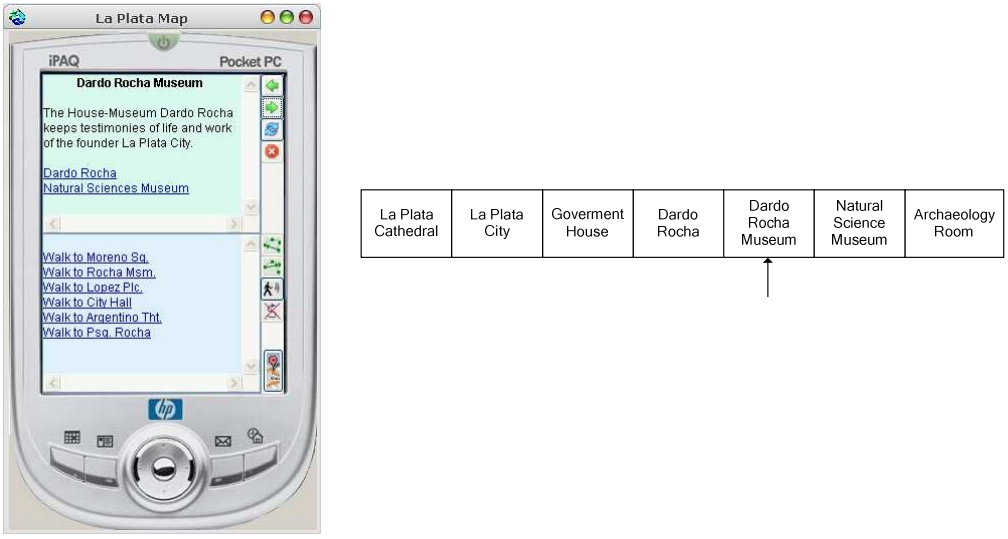


Figura 52: Vista digital después de hacer un next.

Caso 3: El usuario clickea en link digital.

Siguiendo con el ejemplo presentado en la Figura 52; el usuario decide clickear en

el link digital “Dardo Rocha”. En la Figura 53 se ve la nueva página seleccionada por el usuario. Este caso deja en evidencia una de las desventajas de la política *StackBasedModel*, y es que se pueden perder, dentro del historial de navegación, nodos anteriormente visitados.

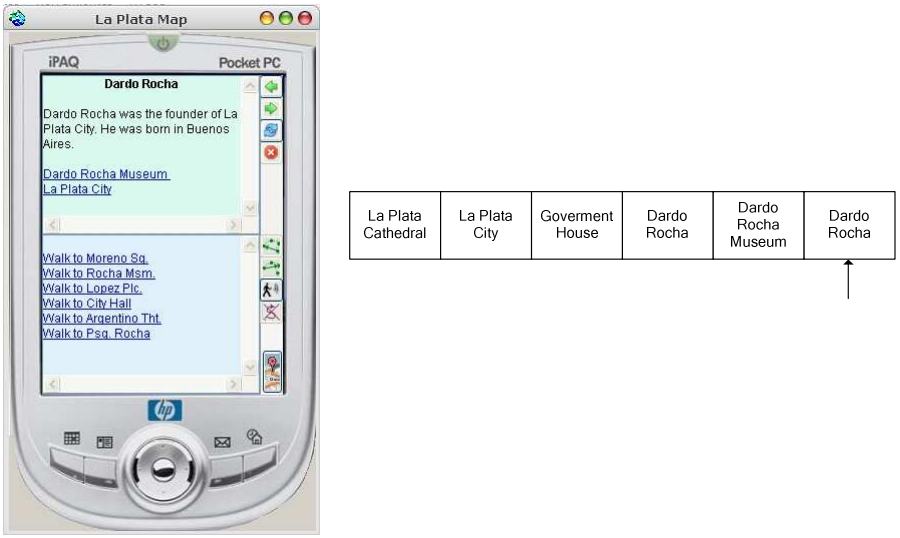


Figura 53: Vista digital después de clickear en link digital.

5.2.3. El usuario hace back, next y clickea en link digital con política *RecencyBasedModel.*

Tomando como base el historial de navegación digital presentado en la Figura 50, pero se usará la política *RecencyBasedModel,* si el usuario hace tres veces back digital podrá ver una pantalla similar a la presentada en la Figura 51 y el historial mantendrá la misma estructura. Si hace una vez next puede visualizar una pantalla como se mostro en la Figura 52. Es decir, después de estas operaciones el usuario esta visualizando la página "Dardo Rocha Museum". Supongamos que decide clickear en el link digital “Dardo Rocha”, en la Figura 54 se ve la nueva página seleccionada por el usuario. Este caso muestra como, a diferencia de la política *StackBasedModel*, no se pierde ningún nodo anteriormente visitado.

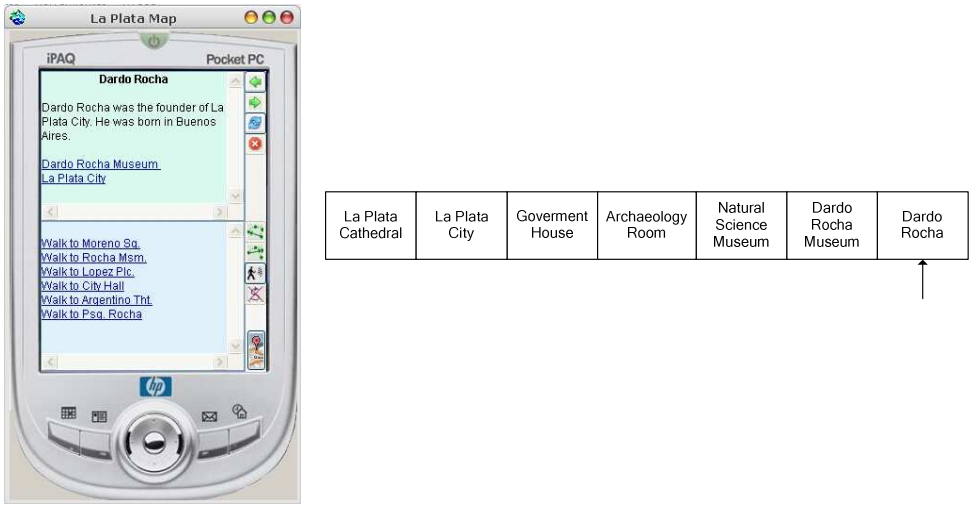


Figura 54: Vista digital después de clickear en link digital.

5.3. Ejemplo Físico

En cuanto a las operaciones posibles dentro de una navegación física son clickear en un link fìsico, hacer back, next, reload o stop.

Supongamos que el usuario fue sensado, después de visitar varios puntos de interés (POI), por el punto de interés "Lopez Plc.". El historial de navegación físico se puede ver en la Figura 55.

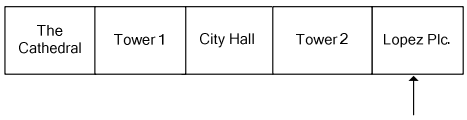


Figura 55: Historial de navegación física.

5.3.1. Usuario caminando - sensado por un punto de interés.

El usuario sigue caminando y es sensado por el POI "Rocha Msm.". Una vez que un POI sensa al usuario, se actualiza la información del browser (tanto lo digital como lo físico). Esto se apreciar en la Figura 56.

Figura 56: Vista de la PDA – frente a "Rocha Msm.".

Si el objeto que sensa al usuario, cuando éste se encuentra caminando, es un objeto físico puro (PO), no se modifica la vista del browser de la PDA; el modelo simplemente detecta el sensado pero no realiza ninguna acción.

5.3.2. Usuario inicia una nueva navegación física (clickear en un link).

Siguiendo con el ejemplo, ahora el usuario se encuentra en "Rocha Msm." y decide visitar, por medio de click en link físico, "Psg. Rocha"; en ese momento se inicia una nueva navegación física a dicho objeto. Una vez que el usuario hizo click sobre un link físico, se actualiza la vista física del browser indicando los datos de la nueva navegación (origen, destino y recorrido sugerido). En la Figura 57 se muestra el resultado.

Figura 57: Vista de la PDA – navegacion fisica a "Psg. Rocha".

Cabe aclarar que los objetos físicos puros (semáforo) que se encuentran dentro del recorrido sugerido serán transparentes al usuario; es decir, el usuario solo verá dentro de los objetos que formen parte de un recorrido sugerido, puntos de interés.

5.3.3. Usuario navegando - sensado por un POI.

Si un POI sensa un usuario que se encuentra navegando, se pueden dar tres casos diferentes. Un punto en común de estos tres casos posibles es que, en la PDA del usuario, se actualizará la información digital con la contraparte digital del punto de interés que lo sensó.

Caso 1: El usuario es sensado por un POI que forma parte del recorrido.

Continuando con el ejemplo de la Figura 57, el usuario es sensado por "Legislatura". En la Figura 58, se ve como se actualiza la información digital y física. En la parte física se indica al usuario que está dentro del recorrido sugerido.

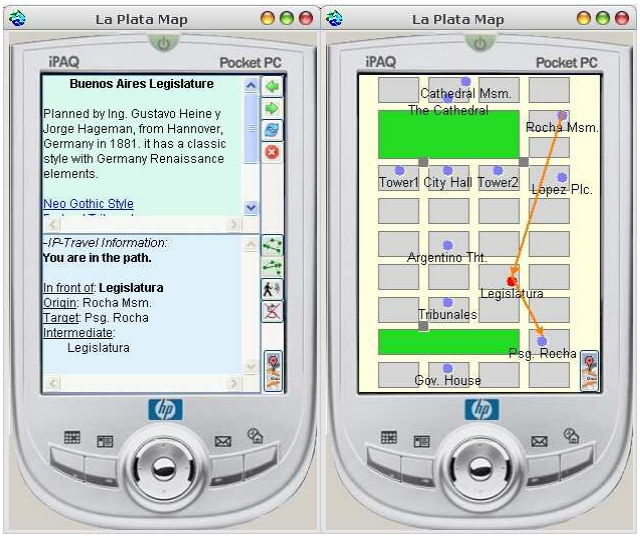


Figura 58: Usuairo sensado por "Legislatura".

Caso 2: El usuario es sensado por un POI que no forma parte del recorrido.

El usuario continúa su camino pero en vez de ir hacia “Psg. Rocha” se equivoca de dirección siendo sensado por “Tribunales”. En la parte física, se le indica al usuario que se encuentra fuera del recorrido. Esto se ve en la Figura 59.

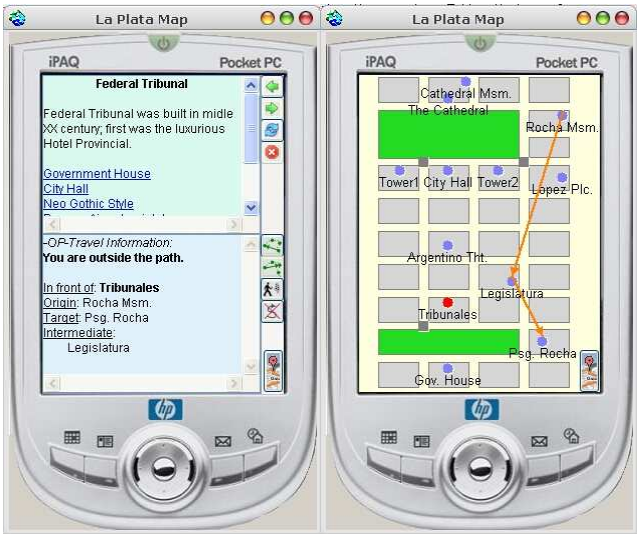


Figura 59: Usuairo sensado por “Tribunales”.

Caso 3: El usuario es sensado por el POI destino.

El usuario sigue camininando hasta que lo sensa “Psg. Rocha”, en ese caso, el POI que sensó al usuario es el target de la navegación, el usuario llegó al final de la misma. La visualización en pantalla de esta acción es similar a cuando un POI sensa un usuario que se encuentra caminando; la única diferencia es que se informa al usuario que llegó al destino elegido (ver Figura 60).

Figura 60: Usuairo sensado por el target “Psg. Rocha”.

5.3.4. Usuario navegando - cancela la navegación.

Un usuario puede cancelar una navegación en cualquier momento. En caso de decidir tal acción, se cancela el recorrido y se muestra en pantalla la información del punto de interés actual que lo sensó.

Caso 1: Usuario frente a un POI – cancela la navegación.

Para ayudar a una mejor comprensión de este ejemplo, rebobinamos el ejemplo hasta el Caso 1 de 5.3.3 (El usuario fue sensado por un POI, Figura 58). Al momento de cancelar una navegación, el usuario se encuentra en “Legislatura”, En la Figura 61 se muestra el estado de la PDA después de cancelar la navegación.

Figura 61: Usuario frente a un POI - Navegación cancelada.

Caso 2: Usuario frente a un semáforo – cancela la navegación.

En caso que un usuario se encuentre frente a un semáforo y decida cancelar su navegación se le indicará en pantalla que no se encuentra frente a ningún POI. En la Figura 62 se muestra el estado de la PDA después de cancelar la navegación.

Figura 62: Usuario frente a un semáforo - Navegación cancelada.

5.3.5. Usuario navegando - sensado por un PurePO.

En 5.3.3 se vió como se actualiza la pantalla de la PDA del usuario (cuando se encuentra navegando) al ser sensado por un POI. A continuación se muestra lo mismo pero para objetos físicos puros, mas precisamente se instancio la aplicación con semáforos.

Un semáforo que sensa al usuario, cuando éste se encuentra navegando, puede o no formar parte del recorrido. Como se vió en 5.3.2, los objetos físicos puros no figuran como posibles nodos intermedios a visitar dentro de una navegación. El objetivo de tales es, simplemente, asistir al usuario informándole si se encuentra (o no) dentro del recorrido.

Caso 1: El usuario es sensado por un semáforo que no forma parte del recorrido.

Continuándo con el ejemplo, el usuario ahora decide iniciar una navegación, desde "Rocha Psg.", a "Argentino Tht.". En camino hacia su target es sensado por un semáforo que se encuentra fuera del recorrido sugerido. En Figura 63 se ve como queda la vista del browser.

Figura 63: Usuario sensado por un semáforo fuera del recorrido sugerido.

A manera de verificar el correcto funcionamiento de la aplicación, se muestran, en pantalla, algunos datos respecto a los nodos; en este caso "*PurePO-OP-Travel Information:*" sirve para indicar que el nodo es un Objeto Físico Puro. Esto será invisible en la aplicación final.

Caso 2: El usuario es sensado por un semáforo que forma parte del recorrido.

A manera de visualizar un ejemplo en el cuál un semáforo se encuentre dentro del recorrido sugerido se reutilizará el ejemplo de inicio de navegación de 5.3.2.

En camino a "Rocha Psg." el usuario es sensado por el semáforo, ubicado en 12 y 50, el cuál se encuentra dentro del recorrido. Como los semáforos no poseen contraparte digital, el sensar un usuario simplemente actualiza la información física

indicando al usuario que se encuentra dentro del recorrido sugerido. A continuación, en la Figura 64 se ve como se actualizó la parte física.

Figura 64: Usuario sensado por un semáforo dentro del recorrido sugerido.

En la parte física de la Figura 64 se puede ver que, si bién el semáforo que sensó al usuario no figura en pantalla como nodo intermedio, pertenece al recorrido sugerido pero esto es invisible al usuario.

5.3.6. Usuario caminando - hace back.

Previo a dar el ejemplo, se ubica al usuario en un contexto de navegación; la Figura 65, extraída de 5.3.1, muestra el historial de navegación física del usuario.

The Cathedral	Nav to Lopez Plc.	Rocha Msm.	Lopez Plc.	Nav to Argentino Tht.	Legislatura	Argentino Tht.	City Hall	Tower 1
PON	STN	IIPTN	RT	STN	IIPTN	RTN	PON	PON

Figura 65: Usuairo sensado por un semáforo dentro del recorrido sugerido.

A continuación, la Figura 66 se muestra la configuración de la política física elegida por el usuario.

Figura 66: Configuración de la política de navegación física.

Según el historial de la Figura 65, el usuario se encuentra frente al “Tower 1”, en ese momento hace tres back seguidos y, posteriormente, un next. A continuación se muestra, en la Figura 67, lo que fue viendo el usuario en el mapa mientras iba haciendo los back y next.

Figura 67: Secuencia de backs y next.

5.3.7. Usuario navegando (back) - sensado por un POI.

Continuándo con el ejemplo de la Figura 67, el usuario inició una navegación, por medio de next, al punto de interés “Argentino Tht.”. En busca de su target, el usuario fue sensado por “City Hall”, “Tower 2” hasta que, finalemente arribó a “Argentino Tht.”. A continuación se muestra, en la Figura 68, lo que el usuario vió en pantalla de la PDA al ser sensado por estos tres puntos de interés.

Figura 68: Secuencia de POI visitados.

5.3.8. Usuario navegando (back) - sensado por un PurePO.

Como introducción, en la Figura 69, se muestra una nueva navegación física iniciada por el usuario, desde “Gov. House” a "City Hall". El semáforo ubicado en 7 y 53 (esquina de “Tribunales”) se encuentra, implícitamente, dentro del recorrido sugerido.

Figura 69: Navegación física a “City Hall”.

A manera de resumir el ejemplo, se supone que el usuario, mientras iba caminando en busca del destino elegido (“City Hall”), fue sensado, entre otros, por los semáforos ubicados en “7 y 53” (dentro del recorrido sugerido) y en “12 y 53” (fuera del recorrido sugerido). Por lo tanto, dentro de esta navegación, el usuario fue sensado por dos objetos físicos puros (semáforo), uno dentro y otro fuera del recorrido sugerido. En la Figura 70 se muestran cuatro pantallas, las primeras dos muestran la información que vió el usuario al ser sensado por el semáforo que se encontraba dentro del recorrido y las otras dos restantes, la información al ser sensado por el semáforo que se encontraba fuera del recorrido.

Figura 70: Sensado de semáforo dentro y fuera del recorrido sugerido.

En las imágenes de las vistas del browser de la Figura 70 se ve como, al ser sensado por un semáforo, se actualiza solamente la parte física debido a que éstos no poseen contraparte digital. Simplemente le indica al usuario que se encuentra frente a un semáforo y que está en el camino correcto y fuera de él.

5.3.9. Usuario navegando (back) - hace back.

Para este ejemplo se rebobinará a la navegación iniciada en la Figura 67 del ejemplo 5.3.6 (el usuario inició una nueva navegación por medio de back, desde "Tower 1", al punto de interés “Argentino Tht.”). Inicialmente el usuario camina, siguiendo el recorrido sugerido, en dirección a "City Hall". Una vez sensado por "City Hall" se actualiza la información del browser (ver Figura 71).

Figura 71: Sensado de “City Hall” en una navegación iniciada por back.

El usuario decide hacer back sin previa cancelación explícita de su navegación actual. En ese momento, se cancela la navegación actual de manera trasparente al usuario y se inicia, de acuerdo a la configuración de su política (ver Figura 66), una nueva navegación física al punto de interés "Tower 1". Esto se puede ver en la Figura 72.

Figura 72: Nueva navegación física, iniciada por back, a “Tower 1”.

A travez de los ejemplos se puede visualizar que las dos navegaciones trabajan en forma independiente. El unico cambio que afecta a las dos navegaciones es el sensado, el mismo se simulo mediante una lista de los objetos físicos, y al cliquear en uno de ellos se simula el sensado, esto dispara la cadena de mensajes para que se actualicen las dos navegaciones.

En esta sección se presentaron ejemplos de navegación digital y física, avarcando la mayor cantidad de casos posibles para mostrar el funcionanamiento del Browser de Hipermedia Física. Para mayor entendiemiento visual se dejo en la pantalla física la descripción de la accion que esta realizando el usuario, ya que esto permite controlar y visualizar mejor el funcionamiento del Browser.
No se detallaron más ejemplos de configuraciones físicas debido a que no cambia la visualizacion que recibe el usuario (en cuanto a contenido visual), sino que se diferencia en los nodos que son considerados para hacer back y next (esto se detallo con ejemplos en el Cápitulo 4).

Capítulo 6. Conclusiones.

Al principio de esta tesis, en el Capítulo 1, se enunciaron los objetivos a cumplir:

- En primer lugar, desarrollar la funcionalidad de navegación de un browser para aplicaciones de Hipermedia Física.
- En segundo término, y como complemento de la parte principal, diseñar políticas personalizadas de navegación.

El primer objetivo se logró por medio de una extensión del modelo abstracto presentado en al Capítulo 3 ; esta extensión dió como resultado el modelo de navegación para un browser de Hipermedia Física (ver Figura 46). El mismo quedó representado por dos modelos independientes, el modelo digital y el modelo físico. Cada Browser tiene su propia barra de navegación, los botones de la barra delegan en su modelo (digital o físico) la resolucion de los mismos permitiendo la actualización del nodo actual. De esta manera se logra una clara separacion entre el Browser y su modelo de navegación.

Para cumplir el segundo objetivo se diseñaron nuevas políticas perzonalidas de navegación, tanto digital como física. Dentro de la parte digital se desarrollaron dos políticas, la Stack-Based y Recency-Based. Es posible agregar nuevas políticas de navegación al digital dado que el modelo de navegación digital es independiente de la vista del browser. Para agregar una nueva política basta con crear una nueva clase como subclase de *NavigationStrategy*. En cuanto a la funcionalidad, una nueva política de navegación aportará una nueva manera de revisitar nodos, por medio del Back y Next. Una nueva política solo necesita redefinir los métodos indicados en la clase abstracta *NavigationStrategy* (como se mostró en 4.1.3).

Para la navegación física se desarrolló una nueva política de navegación configurable, *ConfigPhysicalStrategy*. Dentro de esta política existen dos grupos de configurabilidad: objetos de interés y opciones de navegabilidad; estas últimas opciones son un conjunto de variables que el usuario setea según sus preferencias. Cada una de estas variables representa el interés del usuario sobre los nodos. Esta política resulta adapatable ante posibles nuevos requerimientos; tal adaptabilidad será posible llevarla a cabo sin demasiados cambios. Por ejemplo, supongamos que se quisiera agregar un nuevo estado a los estados de un usuario en un Sistema de Hipermedia Física (Capítulo 3.2). La manera de integrarlo al modelo sería insertar un nuevo nodo como subclase de *PhysicalNode* y definir una nueva variable dentro de la política *ConfigPhysicalStrategy*. Para finalizar, habrá que crea el template con el objetivo de poder mostrar en pantalla la información acorde a dicho nuevo nodo.

La politica física permite que el back y next físico se ajusten a la configuración actual permitiendo brindar funcionalidad acorde a las preferencias del usuario. La configuración de la política puede variar en forma dinamicanmica mientra el usuario esta utilizando el sistma. Esto lo puede realizar el usuario en forma manual o podria tener un sistema que la adapte de acuerdo al comportamiento del usuario.

En esta tesis no se abarco la parte de representación grafica del mapa (como puede ser mapas como Google Maps) ni la realización de cálculo de caminos

reales, ya que este tema estaba fuera del foco de la tesis. Esta seria una posible continuación de la tesis, incorporando mapas y calculo de caminos reales.

Otro posible trabajo futuro podría ser extender los dominios presentados (Navegación Digital y Física) incorporando nuevos dominios navegacionales. Podría darse por ejemplo, el caso de tener un cartero que quiere tener una aplicación de hipermedia física acorde a su tarea particular (repartir cartas). En este caso las políticas además tendrán que considerar esta actividad para determinar que significa por ejemplo hacer back. Podría entenderse hacer back como volver a las casas donde no había nadie para entregar la carta. Este tipo de extensiones (o trabajos futuros) son simples de incorporar ya que solo necesitan definir el modelo navegacional junto con sus políticas. Y de ser necesario definir un nodo acorde a la representación que necesite cada dominio.

También se podría pensar en extender la funcionalidad del browser, como podría ser permitir tener bookmarks. Para lo cual se necesita por un lado extender el modelo general para soportar esta funcionalidad y por otro lado que significa tener bookmarks en cada domino.

Capítulo 7. Referencias Bibliográficas.

[Balzer et al, 1989] R. Balzer, M. Begeman, P. Garg, M. Schwartz, B. Shneiderman. “Hypertext and Software Engineering”. Hypertext 1989. pp.395-396.

[Bieber et al, 1994] M. Bieber, W. Jiangling. “Backtracking in a Multiple-window Hypertext Environment”. In Proceeding of ECHT 1994, pp. 158–166. ACM Press, New York, 1994.

[BMW, 2008] http://www.bmw.com/com/en/owners/service/augmented_reality_introduction_1.html

[Bolter, 1991] J. Bolter. "Writing Space: The Computer, Hypertext, and the History of Writing". Lawrence Erlbaum Associates. 1991.

[Bouin et al, 2003] N. Bouvin, B. Christensen, K. Grønbæk, F. Hansen. “HyCon: A Framework for Context-aware Mobile Hypermedia “. 2003.

[Challiol et al, 2006] C. Challiol, G. Rossi, S. Gordillo, V. De Cristófolo. "Systematic Development of Physical Hypermedia Applications". 2006.

[Challiol et al, 2007] C. Challiol, A. Fortier, S. Gordillo, R. Laurini, A. Muñoz, G. Rossi. “Browsing Semantics in Context-Aware Mobile Hypermedia”. Dans CAMS 07, 3rd International Workshop on Context-Aware Mobile Services, within OTM'2007, November 25-30, Vilam. 2007.

[Challiol et al, 2008] C. Challiol, A. Fortier, S. Gordillo, G. Rossi. “Architectural and Implementations issues for a Context-Aware Hypermedia Platform”. Journal of Mobile Multimedia, Vol 4, No. 2, 2008, 118-138.

[Cockburn et al, 1999] A. Cockburn, S. Greenberg. “Getting Back to Back: Alternate Behaviors for a Web Browser’s Back Button”. In Proceeding of the 5th Annual HFWeb, 1999.

[Cockburn et al, 2002] A. Cockburn, B. McKenzie, M. Jasonsmith. "Pushing Back: Evaluating a New Behaviour for the Back and Next Buttons in Web Browsers". 2002.

[Conklin, 1987] J. Conklin. "Hypertext: An Introduction and Survey". IEEE Computer, Septiembre 1987. pp. 17-41

[Díaz et al, 1996] P. Díaz, N. Catenazzi, I. Aedo. "De la Multimedia a la Hipermedia". RA-MA Editores, Madrid, 1996.

[Fiderio, 1988] J Fiderio. "A Grand Vision--Hypertext mimics the brain's ability to access information quickly and intuitively by reference". Byte Magazine, Vol. 13, Nº 10. October 1988. pp.237-244.

[Fortier et al, 2005] A. Fortier, S. Gordillo, G. Rossi. "Decoupling Design Concerns in Location-Aware Services". In Proceedings of the IFIP Conference on Mobile Information Systems (MOBIS), Springer, 2005.

[Fortier et al, 2007] A. Fortier, C. Challiol, G. Rossi, S. Gordillo. "Physical Hypermedia: a Context-Aware approach". 2007

[Galbreath, 1992] J. Galbreath. "The Educational Buzzword of the 1990s: Multumedia, or is it Hypermedia, or later active Multimedia". Educational Technology. The Magazine for Managers in Education, 32 (4), 15-19.1992.

[Gamma, 1995] E. Gamma, R. Helm, R. Johnson, J. and Vlissides. "Design Patterns: Elements of Reusable Object-Oriented Software ". 1995.

[Goble et al, 2004] C. Goble, S. Harper, S. Pettitt. “proXimity: Walking the Link”. In Journal of Digital Information, Volume 5, Issue 1, Article No 236, 2004.

[Greenberg et al, 1999] S. Greenberg, A. Cockburn. "Getting Back to back: Alternate Behaviors for a Web Browser's Back Button". 1999.

[Greenberg et al, 2000] S. Greenberg, G. Ho, S. Kaasten. “Contrasting Stack-Based and Recency-Based Back Buttons on Web Browsers”. Department of Computer Science, University of Calgary, Calgary, Report 2000-666-18..

[Grønbæk et al, 2003] K. Grønbæk, J. Kristensen, P. Ørbæk, M. Eriksen. “Physical Hypermedia: Organising Collections of Mixed Physical and Digital Material". In Proceedings of Hypertext 2003, 2003, pp. 10-19.

[Hansen et al, 2004] F. Hansen, N. Bouin, G. Christensen, J. Gagach, K. Grønbæk, T. Pedersen. “Integrating the Web and the World: Contextual Trails on the Move”. Proceedings of the fifteenth ACM Conference on Hypertext and hypermedia, 2004.

[Landow et al, 1991] G. Landow, P. Delany. "Hypermedia and Literary Studies". Cambridge: Massachusetts Institute of Technology Press, 1991.

[Lynch, 1991] Lynch. "Tecnología Multimedia. Multimedia, primeros pasos. Guía Apple para Educación". 1991.

[Mackay, 1998] W. Mackay. “Augmented Reality: Linking real and virtual worlds A new paradigm for interacting with computers”. 1998.

[Milgram et al, 1994] P. Milgram, F. Kishino. “A taxonomy of Mixed Reality Visual Displays”. 1994.

[Milic-Frayling et al, 2004] N. Milic-Frayling, R. Jones, K. Rodden, G. Smyth, A. Blackwell, R. Sommerer. "SmartBack: Supporting Users in Back Navigation". 2004

[Nelson, Theodor H. 1965] T. Nelson. "A File Structure for the Complex, the Changing and the Indeterminate". Proceedings of the ACM 20th National Conference, 1965, pp. 84-100.

[O'Connell, 2005] P. O'Connell. “Korea's High-Tech Utopia, Where Everything Is Observed”. 2005.

[Rada, 1991] R. Rada. "Hypertext: from text to expertext". McGraw-Hill. 1991.

[Rossi et al, 2006] G. Rossi, S. Gordillo, C. Challiol, A. Fortier. “Context-Aware Services for Physical Hypermedia Applications”. In Proceedings of the CAMS 2006, Springer-Verlag Berlin Heidelberg, 2006, pp. 1914-1923.

[Schlumpf, 1990] Schlumpf. "Multimedia Applications in MSDOS". The Seventh International Conferencie on Technology and Education (I). Edinburg: CEP Consultants. pp. 95-97.

[Tauscher et al, 1997] L. Tauscher, S. Greenberg. "How people revisit web pages: empirical findings and implications for the design of history systems". 1997

[Veljkov, 1990] citado en F. Blázquez, J. Cabero y otros. “Nuevas tecnologías de la Información y Comunicación para la Educación”. Sevilla, Alfar, 1994.

[Wagner et al, 2005] D. Wagner, T. Pintaric, F. Ledermann, D. Schmalstieg.”Towards Massively Multi-User Augmented Reality on Handheld Devices”. 2005.

Anexo A: Templates.

Como se vió en la sección 4.2.2, *PhysicalNode* define el método `html` el cuál es utilizado por los nodos para mostrar su información en pantalla. Cada nodo tendrá su template correspondiente con el objetivo de poder mostrar en pantalla información acorde al mismo. A continuación se muestra, a manera de ejemplo, el template de *PointOfInterest*.

Template *PointOfInterest*

```
<html>
<head></head>
<body>
      <% document include: 'links' target: target %>
</body>
</html>
```

Template *links*

```
<%
|physicalObject|
physicalObject := target.
document write: '<br><br>'.
physicalObject physicalLinks do:[:p |
      document write: '<a href="physicalLink='.
      document write: (p target name copyWithout: Character
      space).
      document write: '"> Walk to '.
      document write: p target name.
      document write: '</a><br>'.].
%>
```

Código 43: Template `PoinOfInterest` y `links`.

El template `links` se encarga de generar el código html para los links físicos de un *PhysicalNode* generando, por cada link, un hipervínculo.

A continuación se muestra el método `html` (y submétodos) de *PhysicalNode*:

```
html
 ^self htmlFor: self targetNode.
```

Código 44: Método `html` de PhysicalNode.

El Código 44 muestra que se invoca al método `htmlFor:` (el cual se encuentra definido en el Código 45).

```
htmlFor: aTargetNode
 | paths |
 paths := aTargetNode class withAllSuperclasses collect: [:cls
 | self path constructSafe: (Filename separator asSymbol
 asString , cls name)].
 ^self buildHTMLFrom: (paths detect: [:filename | filename
 exists]) target: aTargetNode.
```

Código 45: Método `htmlFor:aTargetNode` de PhysicalNode.

El Código 45 muestra que se generan posibles nombres de archivos a partir de las subclases (`paths`), el cual sirve para buscar el archivo template asociado al nodo (`aTargetNode`). La busqueda del template la realiza el método `buildHTMLFrom:aFilename target:anObject` (el cual se visualiza en el Código 46).

```
buildHTMLFrom: aFilename target: anObject
  | bindings document |
  bindings:=Dictionary new.
  bindings at: 'target' put: self targetNode.
  document:=(Template from: aFilename) processInto:
  (TargetDocument inMemory: 'HTML') extraBindings: bindings.
  ^document text.
```

Código 46: Método `buildHTMLFrom:aFilename target:anObject` **de PhysicalNode.**

En el Código 46 se puede visualizar que se toma el template indicado por `aFilename` y al cual se le pasa `anObject` para que lo pueda mostrar. En el caso de que el nodo a visualizar sea *PointOfInterest* el template que se procesa será el presentado en el Código 43. El target que le llega al template es el nodo (`targetNode`) y de este target se extraen los datos a visualizar. Una vez procesado el template se obtiene el html el cual sirve para ser mostrado en el Browser.

Anexo B: WithStyle.

De manera de poder embeber un browser en una ventana, se utilizó WithStyle. La ventana tiene un objeto de la clase WithStyle.Client.WsRenderWidget, y este objeto es el encargado de visualizar la información en html.

Para el desarrollo se utilizaron dos browser, uno para la ventana digital (*DigitalWebView*) y otro para la ventana física (*PhysicalWebView*). La información, en ambos browser, se actualiza por medio del método `updateWebPage`.

```
updateWebPage
   ^self openURL: sel model currentNode html.
```

Código 47: Método `updateWebPage` de DigitalWebView.

Para poder actualizar el contenido del browser, se le envía a este objeto el mensaje `openURL:aString`, donde `aString` será código html generado por el método `html` (ver código 44 de Anexo A: Templates). Acá se ve como, por medio de templates, se genera el código html que luego será interpretado por el browser, tanto digital como físico.

El mensaje `updateWebPage` se dispara cada vez se actualiza el `currentNode` de cada modelo, tanto *DigitalBrowserModel* como *PhysicalBrowserModel*. A continuación, en el código 48 se muestra el método `model:DigitalBrowserModel` de *DigitalWebView*.

```
model: aModel
   self model ifNotNil: [self model retractAllInterestsFor:
   self].
   self setModel: aModel.
   self model ifNotNil: [self model expressInterestIn:
   #currentNode for: self sendBack: #updateWebPage].
```

Código 48: Método `model:DigitalBrowserModel` de DigitalWebView.

Por ejemplo, cuando el usuario clickea sobre un link digital, el *DigitalBrowserModel* actualiza su `currentNode`, el cuál dispara implícitamente el mensaje `updateWebPage` a *DigitalWebView* originando una actualización de la vista o parte digital con el nuevo objeto digital seleccionado.

Los métodos `updateWebPage` y `model:` de *DigitalWebView* son similares en *PhysicalWebView*, la única excepción es que al método `model:` se le pasa un *PhysicalBrowserModel*.

Por lo tanto, se utiliza el componente de WithStyle simplemente como parte de la visualización para mostrar los datos de los nodos. Es decir, se mantiene la estructua abstracta presentada en el Cápitulo 3 donde el Browser delega en su modelo la resolución de la actualización de los nodos y simplemente muestra lo que le devuelve su modelo. En este caso el componente de WithStyle visualiza el html asociado al nodo actual.

No se utilizan requerimientos http, la combinación de los templates junto con el modelo asociado a cada Browser permiten la actualización de las pantallas del usuario. Esto permite independencia en la visualización ya que el componente de WithStyle puede ser reeemplazado por cualquier otro componente que permita la visualización de texto html.