4. Modelo de Proceso Flexible de Hipermedia como soporte al desarrollo de Aplicaciones

El objetivo de este capítulo consiste en describir, en un nivel de granularidad media, distintos aspectos de un proceso integral de desarrollo de artefactos de hipermedia. Para una mejor comprensión de esta parte del trabajo, lo estructuraremos del siguiente modo:

- primero definiremos de un modo integral, las fases, tareas y un orden parcial de realización de las mismas (perspectiva funcional). Principalmente nos concentraremos en la fase de desarrollo del MPFH.

- describiremos y redefiniremos, cuando sea necesario, a constructores de proceso que favorezcan al potencial empleo y equilibrio entre el modelado lógico (formal o semi-formal), y el modelado físico en el proceso de desarrollo de un proyecto de hipermedia. Remitiremos al lector a las distintas fuentes de la literatura investigada.

- propondremos a la prototipación flexible como a una estrategia participativa que, esencialmente, promueve el ciclo de aprendizaje, descubrimiento, construcción, demostración y validación basado en la retroalimentación experimental entre los usuarios y desarrolladores.

- discutiremos la perspectiva de comportamiento, sobre todo en la fase de desarrollo

- comentaremos sobre criterios cognitivos en el desarrollo de artefactos de hipermedia

4.1. Introducción

Debemos distinguir entre un modelo de proceso de software de otro tipo de modelado de procesos porque muchos de los aspectos representados, cuando se instancian en un proyecto real, son controlados y ejecutados por agentes humanos antes que por agentes computarizados. De manera que un modelo de proceso de software trata con aquellos fenómenos de un proyecto que suceden en la generación, evolución, administración y mantenimiento de artefactos de software.

Todo proceso de software, como indicamos previamente, se puede abstraer en un modelo de proceso. Nuestro modelo de proceso flexible abstrae al ciclo de desarrollo en tres fases fundamentales, que, independientemente del tipo de aplicación y el tamaño del proyecto se pueden observar como común denominador de todo proceso de software; estas son, a saber: la fase de definición (llamada de exploración en nuestro MPF), la fase de desarrollo (o también llamada de modelado dinámico) y la fase de mantenimiento (operativa o de vida de los productos).

Estas tres fases, sus respectivas tareas e interrelaciones generales fueron introducidas en la sección 2.4.

En el siguiente punto concentraremos nuestro interés en la fase de desarrollo, para presentar las distintas perspectivas. El modelo de ciclo de vida que hemos desarrollado hasta el presente, es semi-formal y emplearemos enfoques prescriptivos para su representación y diagramas de clases para su ejemplificación. Algunas de las conclusiones que indicaremos respecto de la estrategia de prototipación fueron validadas observacional y empíricamente.

4.2. La Fase de Desarrollo

Desde una perspectiva funcional, describiremos el secuenciamiento parcial de tareas y actividades, en un nivel medio de granularidad; desde una perspectiva informacional, los artefactos producidos en cada tarea y qué objetos cooperan con la entradas y salidas de las tareas; desde un punto de vista metodológico, analizaremos distintos constructores de proceso, centrados en modelos, para ayudar a construir artefactos, documentar y favorecer al modelo de seguimiento.

En la perspectiva de comportamiento, analizaremos esencialmente, para esta fase, porqué el MPF se desarrolla en una combinación de estrategias iterativa, concurrente, incremental y oportunística. Muy pocas referencias realizaremos a la perspectiva organizacional del modelo de proceso.

En la figura 4.1 hemos realizado una división de procesos para la fase de desarrollo teniendo en cuenta el esquema mostrado en la figura 2.1. En esta fase, que es el núcleo del modelado dinámico podemos observar tareas de planificación, ingeniería de requerimientos, control, diseño, construcción, validación, integración, empleo de patrones de diseño, empleo de criterios

cognitivos, aseguramiento de la calidad y documentación. Se hace uso intensivo de modelos lógicos y físicos.

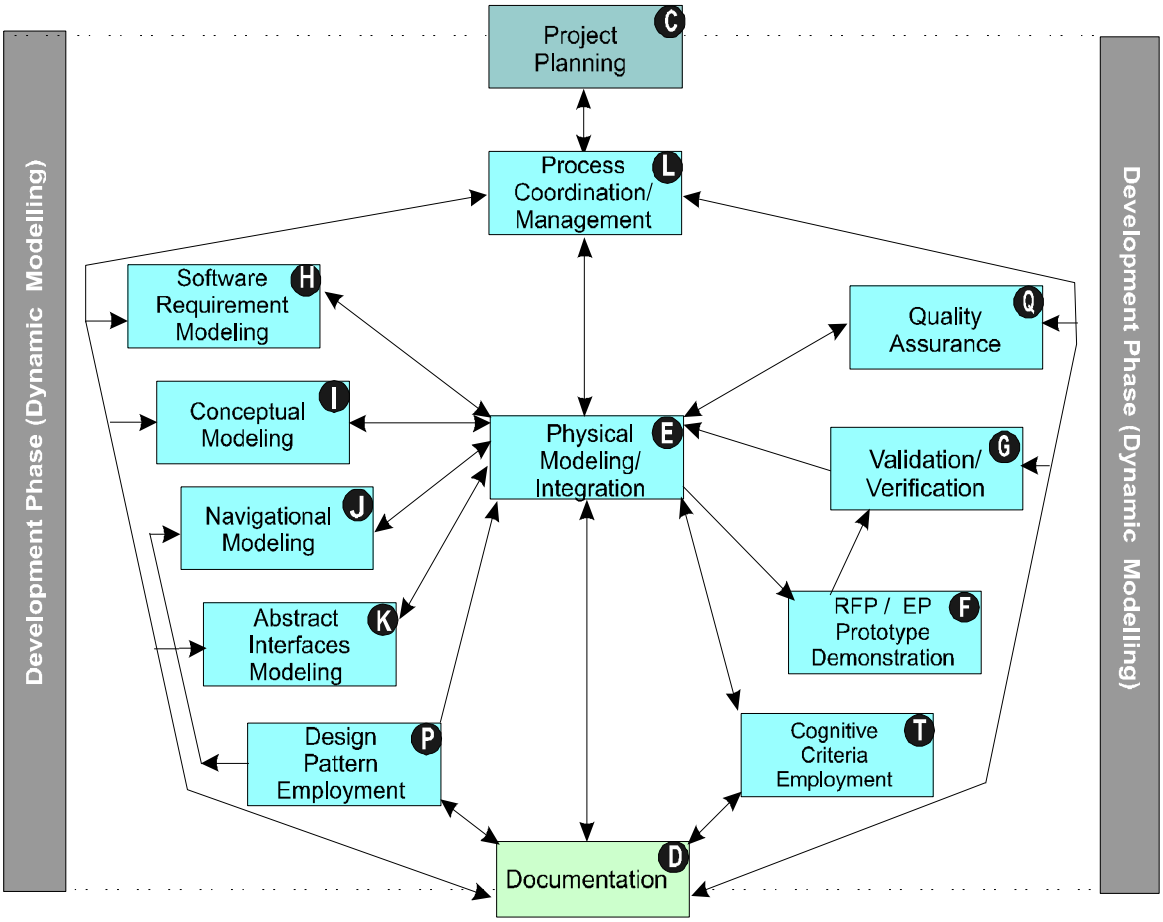


Fig. 4.1 *Principales tareas de la fase de desarrollo*

El centro de operaciones es el proceso L que funciona como el pivote de una actividad global que se la podría asemejar al ciclo de Deming [Deming 86] resumido en estas cuatro acciones: planificar-hacer-chequear-actuar. El rol principal asociado al agente para la realización de esta tarea es el de administrador del proyecto. Esencialmente coordina, esto es, envía y recibe flujos de información y control hacia y desde procesos como planificación, prototipación, modelado conceptual, navegacional y de interfaces abstractas, documentación, validación, testeo y aseguramiento de la calidad entre otros.

Durante cada tarea de nuestro proceso de desarrollo de hipermedia, se está generando o enriqueciendo algún modelo, pudiendo ser algún modelo físico, como algún prototipo de

software o algún sketch en papel, o algún modelo lógico como un diagrama de clases, un diagrama de navegación, etc. Dentro de la fase de desarrollo hay tareas que eminentemente se apoyan en modelos lógicos como son las tareas de modelado conceptual, navegacional y de interfaces abstractas; otras, en una mezcla de modelos lógicos y físicos como las tareas de requerimientos y documentación y, por último, hay tareas que se apoyan principalmente en modelos físicos como el modelo de prototipación.

A seguir, discutiremos distintos aspectos de la fase de desarrollo, principalmente para los procesos H, I, J, K, E, F, G y D. En la figura 4.2 se observa el diagrama de algunos de los modelos lógicos y físicos. Los distintos constructores de proceso emplean modelos concretos para especificar, crear y evolucionar a los artefactos de esta fase, a saber: el modelo de requerimientos así como los modelos conceptual, navegacional, de interfaces abstractas y de prototipación (no discutiremos algunos otros constructores de proceso para tareas de planificación, aseguramiento de la calidad de artefactos, etc.).

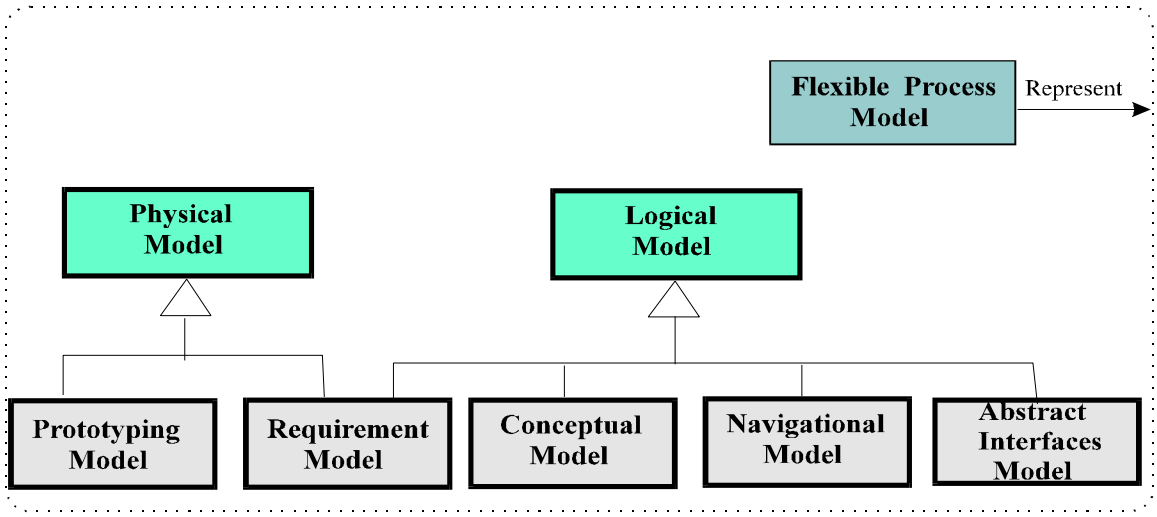


Fig. 4.2 *Algunos modelos lógicos y físicos a utilizar en el proceso de desarrollo*

4.3 El modelado de Requerimientos

Como explicamos en la sección 2.4 la fase de exploración produce una especificación inicial de requerimientos que sirve de entrada a la tarea H, denominada modelado de requerimientos. A partir de esas descripciones de proceso (en una notación informal, como descripciones en lenguaje natural, o con notaciones más formales), nosotros podemos refinarlas y alimentar al modelo de casos de uso, al modelo de glosario y al modelo de interface (figura 4.3). Estos

documentos corresponderán a la salida de la tarea H.

En este trabajo nos detendremos a discutir el modelo de requerimientos, dado que usamos una ligera adaptación para el desarrollo de productos de hipermedia, del bien conocido modelo de casos de uso de Jacobson et al [Jacobson et al 92, Jacobson 94]. Compartimos el punto de vista de los autores que entre los primeros modelos de software está el de requerimientos, ya que describe al sistema, al entorno y sus interacciones, ofreciendo una vista externa del sistema.

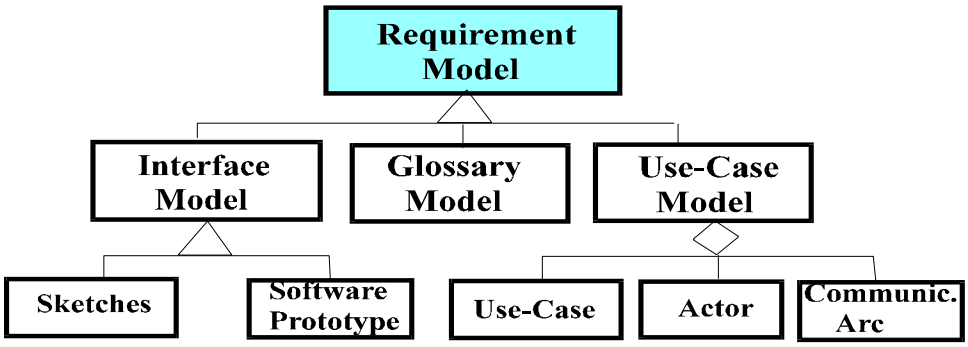


Fig. 4.3 *Tres modelos que conforman al modelo de requerimentos*

El modelo de casos de uso es un modelo lógico que se centra en las interacciones de distintos actores con el sistema; al sistema se lo modela como un conjunto de casos de uso el cual "implementa" toda la funcionalidad esperada del mismo.

En el modelo de glosario se especifican las palabras claves y sus conceptos asociados.

El modelo de interfaces es un modelo que esencialmente usa sketches en papel o soporte semejante y prototipos de software. El modelo de interface captura, básicamente, aspectos funcionales y estéticos en la interacción de los usuarios con el sistema (interface hombre-máquina).

A seguir nos concentraremos principalmente en el modelo de casos de uso, ejemplificando con el Sistema de Información Académico (SIA) introducido en la sección 2.4.1.

4.3.1 El modelo de Casos de Uso

El *modelo de casos de uso* consiste en un grafo con dos tipos de nodos: el nodo actor (comparar con el concepto de agente, de nuestro modelo conceptual) y el nodo caso de uso. Además, tiene

arcos de comunicación.

Al modelo de casos de uso se le asigna un nombre que generalmente corresponde al nombre del sistema. A cada nodo actor se le asocia un nombre y una clase. Por otra parte a cada nodo caso de uso también se le asocia un nombre y un clase los cuales deben ser únicos. Un nodo actor está conectado al menos a un nodo caso de uso y este, a su vez, está conectado al menos a un nodo actor a través de un arco de comunicación (ver figura 4.4).

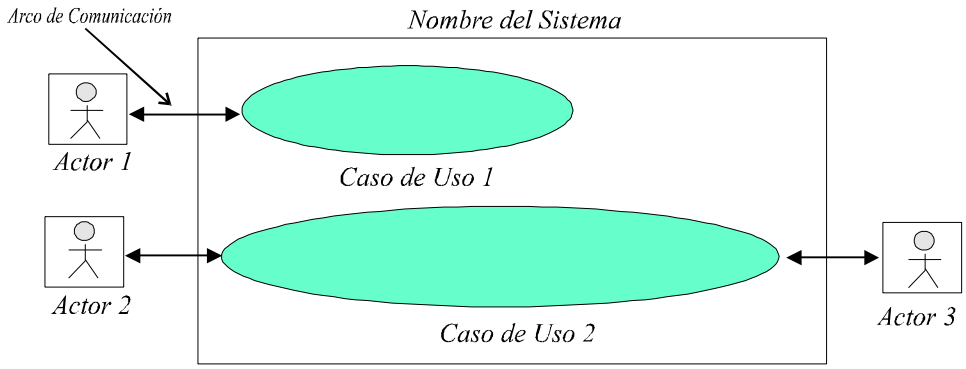


Fig. 4.4 *Esquematización de los elementos del modelo de casos de uso*

Una instancia de un actor puede crear instancias de una clase de caso de uso. Un arco de comunicación entre un nodo actor y un nodo caso de uso implica que se puede enviar un estímulo entre instancias de la clase actor e instancias de la clase caso de uso.

Los actores son objetos que se encuentran fuera del sistema a modelar. Los casos de uso son objetos que se encuentran dentro del sistema. Los actores representan entes externos que tienen necesidad de intercambiar información con el sistema. Los actores pueden ser instanciados por usuarios, dispositivos u otros sistemas. Jacobson et al señalan una diferencia importante entre el concepto de actor y el concepto de usuario, a saber: el concepto de usuario no es formal. El usuario es un ente humano que usa al sistema en tanto que un actor representa un rol específico que el usuario puede jugar. Los actores son instancias de una clase, los usuarios son un tipo de recurso que implementa esa instancia. Bajo este concepto, un mismo usuario puede actuar como instancias de varios actores diferentes, es decir, puede jugar diferentes roles (al igual que en el concepto de agente, rol y recurso presentado en la sección 3.3.1).

Por ejemplo, en el SIA, el rol de auxiliar docente (un actor) y el rol de estudiante (otro actor) pueden ser implementados por un mismo usuario, en ciertas secuencias de uso.

Como se expuso previamente, cuando un actor usa al sistema el mismo realiza un caso de uso. La colección de casos de uso es la funcionalidad completa del sistema. En la figura 4.5 mostramos a modo de ejemplo los casos de uso de un SIA reducido.

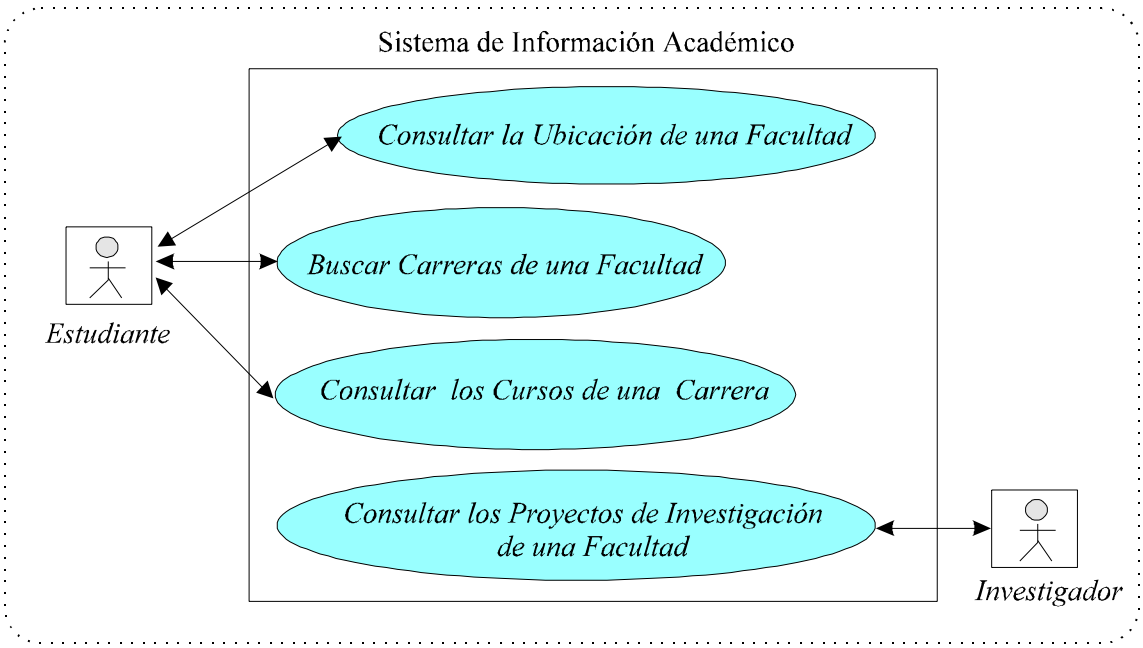


Fig. 4.5 *Casos de uso reducido de un SIA*

Representamos a las clases "*Estudiante*" e "*Investigador*" como nodos actores del modelo; y a las clases "*Consultar la Ubicación de una Facultad*", "*Consultar los Cursos de una Carrera*", etc., como nodos caso de uso, y los correspondientes arcos de comunicación entre nodos actores y nodos caso de uso.

En un momento del proceso de elicitación de requerimientos, un caso de uso puede representar un alto nivel de abstracción. Esto es, en el proceso se puede descubrir que un caso de uso está compuesto por casos de uso anidados. Por ejemplo, el caso de uso "*Consultar los Cursos de una Carrera*" podría componerse por los casos de uso "*Consultar los Cursos de una Carrera por Módulos*" , "*Consultar los Cursos de una Carrera por Departamento*", etc.

En la fig. 4.6 desarrollamos una secuencia del caso de uso "*Consultar los Cursos de una Carrera por Módulos*" en donde podemos apreciar la especificación de un curso de acción normal y cursos alternativos.

El curso normal es el curso de acción básico que da una clara y sencilla comprensión del caso de uso. En cambio un curso alternativo especifica variantes del curso normal o situaciones excepcionales o de error. Normalmente un caso de uso tiene un curso de acción normal y de cero a varios cursos alternativos.

CASO de USO: *Consultar los Cursos de una Carrera por Módulos*

Curso Normal

- ☞ El usuario elige el contexto temático Carreras del menú principal
- ☞ Obtiene un índice jerárquico de Carreras por Facultad
- ☞ Elige una Carrera del índice
- ☞ Obtiene los Cursos de una Carrera para un Módulo:
  Los módulos de una carrera de grado puede estar dividida, por ejemplo, en Módulo Básico, Técnico y de Especialización
- ☞ Obtiene los distintos Módulos:
  Presionando los botones “siguiente” o “anterior” puede obtener los correspondientes módulos.

Curso Alternativo 1

- ☞ El usuario decide retornar al menú principal:
  En todo momento el usuario tiene visible el botón de retorno al menú principal como manera de abortar la secuencia de pasos para acceder a la información

Curso Alternativo 2

- ☞ El usuario decide imprimir el Plan Reducido de la Carrera por Módulos

Fig. 4.6 *Curso Normal y Alternativos de un Caso de Uso particular*

Como se puede apreciar en el ejemplo previo, una instancia de caso de uso es una secuencia de transacciones realizadas por el sistema, la cual produce un resultado medible para un actor en

particular (en nuestro caso el estudiante). Se pude ver a los casos de uso como objetos transaccionales con estado y que se pueden manipular a través de la comunicación con ellos.

4.3.1.1 Comentarios

Finalmente, así como es importante tener en cuenta qué se puede modelar con casos de uso, del mismo modo es relevante considerar qué cosas están fuera del alcance de este modelo de requerimientos (el cual representa, como se explicó previamente, una visión externa del sistema a construir):

1. no se debe modelar interacciones entre casos de uso dentro del sistema. Así las instancias del caso de uso que serían como los únicos objetos del modelo solo se comunicarán con instancias de actores que residen en el contexto.

2. no se deben expresar conflictos entre instancias de caso de uso, ya que, si bien estas relaciones son muy importantes, son detalles internos que no deben expresarse aquí sino en el modelo conceptual (que discutiremos en la sección 4.4).

3. el modelo de casos de usos no expresa concurrencia. Se puede ver a las instancias de cada caso de uso como una serie de transacciones atómicas secuenciales en el tiempo.

4. los casos de uso sólo puede expresar asociaciones entre clases y no entre instancias (este tipo de asociaciones está discutida con amplitud en la literatura citada de Jacobson).

4.3.2 El modelo de Glosario

El *modelo de glosario* sirve para especificar y comunicar en etapas tempranas del proceso de desarrollo, las palabras claves del dominio del problema. Las palabras claves y sus conceptos relacionados se escriben en lenguaje natural, en una lista clasificada. Esta lista de conceptos claves sirve para alimentar a la tarea de modelado conceptual (tarea I) como medio para encontrar clases. Asimismo sirve para alimentar al modelado navegacional (tarea J), como medio para encontrar nodos o contextos navegacionales (ver sección 4.5 y 4.8).

Por ejemplo, a partir del caso de uso de la figura 4.6, identificamos un conjunto de palabras claves del dominio del problema como "*Curso*", "*Carrera*", "*Facultad*", "*Módulo*" y sus definiciones asociadas, que en algunos casos son obvias. No obstante, las palabras claves se pueden ir anotando en tanto se elicitan los requerimientos iniciales con técnicas como entrevistas u otras y no necesariamente a partir de los casos de uso.

4.3.3 El modelo de Interface

El *modelo de interface* usa modelos físicos tales como sketches en papel o en algún medio semejante, y prototipos de software que son de utilidad en el modelado de requerimientos. (La estrategia de prototipación y sus distintas variantes es discutido con mayor detalle en la sección 4.7).

La construcción de estos modelos físicos ayudan a elicitar requerimientos funcionales y estéticos involucrando a usuarios y desarrolladores en etapas tempranas del ciclo de desarrollo.

Un mecanismo para validar el modelo mental de los usuarios en la realización de las tareas es a través de la construcción de prototipos. Un prototipo de software es un modelo físico implementado sobre computadoras que sirve esencialmente para promover el ciclo de aprendizaje, construcción, demostración y validación basado en la retroalimentación experimental entre los participantes. Un prototipo nos permite aprender, descubrir, evaluar y refinar requerimientos funcionales y no-funcionales.

Por lo tanto, una descripción de los artefactos hipermediales en la tarea de requerimientos consistiendo de un conjunto de casos de uso, y eventualmente actores, juntamente con descripciones en papel y prototipos de software, de manera que involucre a usuarios y desarrolladores, puede contribuir a descubrir errores en etapas tempranas de desarrollo.

En aplicaciones de hipermedia es de crucial importancia construir buenas interfaces y estructurar, con criterios de coherencia [Thüring et al 95, Schwabe et al 95a], a las unidades de navegación.

4.4 El modelado Conceptual

Si bien la metodología de diseño de hipermedia orientada a objetos (OOHDM) es de cuatro etapas consideraremos tres de sus procesos en la que se definen los modelos independientes de los entornos de implementación. Ellos son, según se puede observar en la fig. 4.2: el modelo conceptual (que se podría comparar al modelo de análisis del dominio de las metodologías OO), el modelo navegacional y el modelo de interfaces abstractas.

En la tarea de *modelado conceptual* (tarea I de la figura. 4.1) básicamente se producen dos artefactos: el modelo de clases (eventualmente enriquecido con un modelo de instancias), y las tarjetas de especificación, semejantes a las tarjetas CRC [Wirfs-Brock et al 90, Rossi 96a].

El objetivo esencial por el cual se construye el modelo de clases es que sirve como medio para especificar la semántica del dominio del problema. Se utilizan primitivas tales como clases, relaciones y subsistemas (fig. 4.7). A diferencia de otras metodologías OO tradicionales como la de Booch o Rumbaugh [Booch 94, Rumbaugh et al 91], los atributos pueden ser multitipados.

En el proceso de desarrollo del modelo se emplean mecanismos de generalización/especialización, composición y modularización.

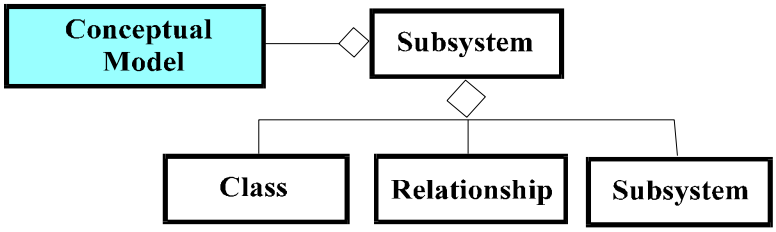


Fig. 4.7 *Diagrama de composición de las primitivas del modelo conceptual*

En cuanto a las tarjetas cumplen un rol muy importante en los procesos de documentación (tarea L) y mantenimiento. Permite especificar, en un estilo textual estructurado, detalles de las partes salientes de clases, objetos, relaciones y subsistemas, facilitando asimismo, la incorporación de información de seguimiento tanto hacia atrás como hacia adelante. Por ejemplo, se podría saber rápidamente observando a la tarjeta, de qué (nombres de) casos de uso proviene una clase (seguimiento hacia atrás -backward traceability), y a qué clases navegacionales alimenta dicha clase del modelo conceptual (seguimiento hacia adelante -forward traceability).

Seguidamente, esquematizaremos con un diagrama de clases, relaciones y subsistemas al modelo conceptual del proyecto *"Facultad de Ingeniería"* [Olsina et al 95].

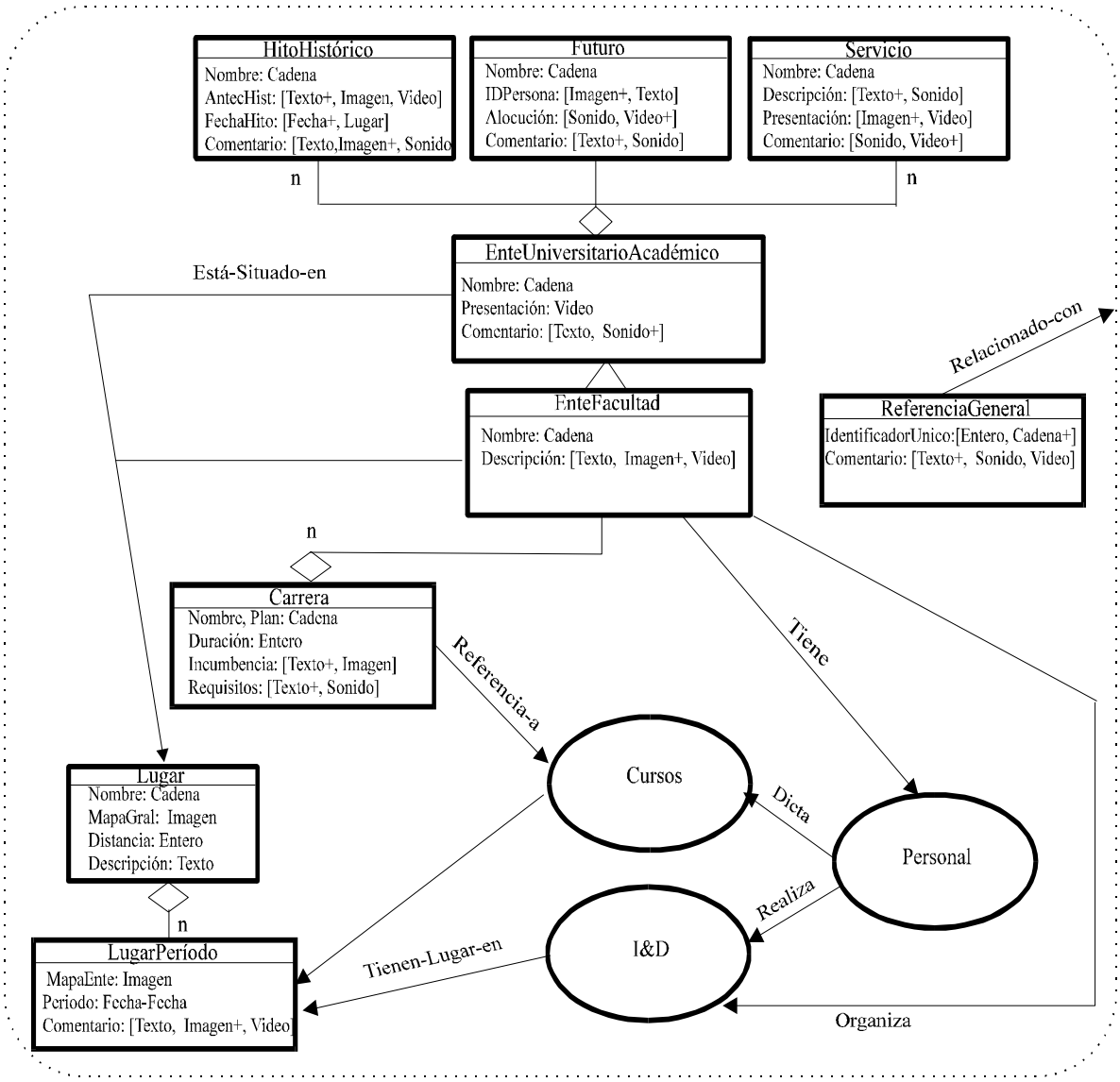


Fig. 4.8 *Diagrama de clases, relaciones y subsistemas del modelo conceptual del SIA*

Como se aprecia en la figura 4.8 algunas relaciones son explícitas y se traducen en enlaces y otras son implícitas. Vemos relaciones de herencia entre las clases "*EnteUniversitarioAcadémico*" y "*EnteFacultad*" y mecanismos de agregación como "*Carrera*", "*Futuro*", con su respectiva cardinalidad. Las clases agregadas son heredadas por las instancias de la clase "*EnteFacultad*". Estos mecanismos favorecen el reuso y la extensión; por ejemplo el ente citado abstrae características y comportamientos comunes de instancias

como "*Facultad de Ingeniería*", "*Facultad de Veterinaria*", "*Facultad de Agronomía*", etc.

Por otra parte se muestra en cada clase los posibles tipos y perspectivas de los atributos. El tipo del atributo puede ser uno primitivo ("*Cadena*"); uno que representa el tipo de objeto multimedial a usar en la aplicación ("*Imagen*", "*Video*"); o una relación implícita como "*Lugar*" en "*HitoHistórico*". El caracter "+" implica la especificación de un tipo por defecto.

(En el modelo conceptual para un atributo se pueden indicar varios tipos, no obstante en la traducción de una clase o clases a un nodo del modelo navegacional, un atributo del nodo puede contener un solo tipo por vez.)

4.5 El modelado Navegacional

Es sabido que hipermedia se caracteriza por el manejo no secuencial de la información y que una característica esencial del diseño es el concepto de navegación. En el proceso de *modelado navegacional* (tarea J) se deben tener en cuenta principalmente aspectos cognitivos (que discutiremos en la sección 4.10), el perfil de usuario para cada vista a construir a partir del modelo conceptual, y los distintos contextos y transformaciones navegacionales.

Las primitivas de construcción son los nodos, los enlaces, las clases y los contextos navegacionales. La dinámica se especifica a través de diagramas de transformaciones navegacionales (ver fig. 4.9). Los nodos pueden ser atómicos o compuestos. Para cada enlace se puede especificar atributos, comportamiento, objeto origen y destino del enlace. también se pueden definir estructuras de acceso y visitas guiadas útiles para encontrar y navegar por espacios de información.

Un contexto de navegación es una primitiva de diseño (patrón de diseño [Rossi et al 97]) y está compuesto de nodos, enlaces, y otros contextos (los cuales pueden estar anidados). Esta primitiva permite representar unidades coherentes de conceptos relacionados y establecer enlaces semánticos apropiados de manera de facilitar la orientación del usuario en dicho espacio [Thüring et al 95]. Se construyen tres esquemas fundamentales en esta etapa: los esquemas de clases y contextos navegacionales y el esquema de transformaciones de navegación. Los dos primeros representan la parte estática de componentes o de una aplicación de hipermedia, en tanto que el segundo especifica la dinámica.

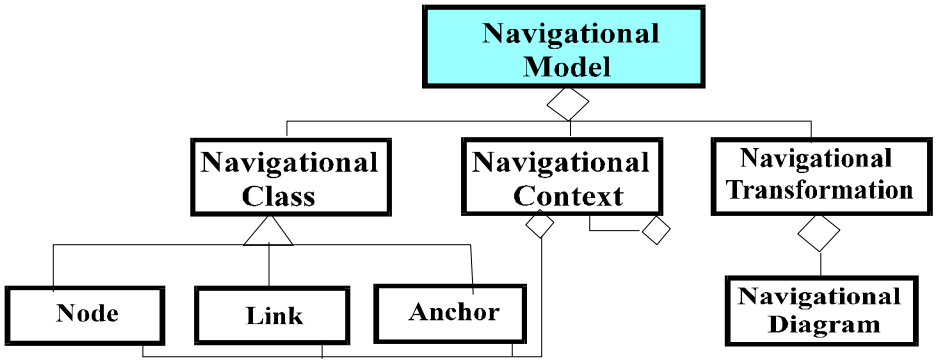


Fig. 4.9 *Principales componentes del Modelo Navegacional*

En cuanto a las tarjetas en esta etapa cumplen un rol semejante al previamente discutido. Permite especificar información de las partes salientes de las clases como nodos, enlaces y contextos de navegación, facilitando la incorporación de información de seguimiento tanto hacia atrás como hacia adelante. Por ejemplo, dada una clase de nodo, se podría saber rápidamente, observando a la tarjeta, de qué clase (o clases) del modelo conceptual proviene, y a qué interfaces abstractas alimenta.

Clase	Carrera	
Atributos: Nombre.Carrera: Cadena Incumbencia: Texto con Anchors Requisitos: Texto con Anchors Plan: Cadena Duración: Entero Contenido: Anchor(Referencia-a) Próximo: Anchor(próximo nodo) Anterior: Anchor(nodo anterior) Arriba: Anchor(a EnteFacultad) Histórico: Anchor(backtracking)		
Relacionado con:	Clase Carrera Carrera Curso	Anchor Proximo Anterior Contenido
Interviene en Contextos Navegacionales: Cursos de una Carrerra, Ubicación, Carrera , Carreras por Facultad		
Parte de: EnteFacultad		
Seguimiento Hacia Atras Carrera en Modelo Conceptual	Seguimiento Hacia Adelante ADV Carrera	

CN	Carrera	Derivado de Clase
Incluye: Carrera: 1..n		
Comportamiento: Paso a paso		
Recorrido: Secuencial, (o acceso directo por medio de un índice contextual)		
Seguimiento Hacia Atras Carrera en Modelo Conceptual	Seguimiento Hacia Adelante ADV Carrera	

Fig. 4.10 *Tarjetas de especificación de la clase nodo y del contexto navegacional denominados“ Carrera”.*

En la fig. 4.10 podemos apreciar a modo de ejemplo una clase nodo como visión del modelo conceptual y uno de los contextos de navegación.

4.6 El modelado de Interfaces Abstractas

La metodología OOHDM diferencia el proceso de construcción del modelo navegacional del proceso de *modelado de interfaces abstractas* (tarea K de la figura 4.1)

Se pueden construir "n" interfaces abstractas para un mismo modelo navegacional (como queda especificado por el enlace "*defineApariencia*" de la fig. 4.14). Los aspectos a tener en cuenta en esta tarea radica en definir qué eventos intervendrán en el lenguaje de acción, qué objetos percibirá el usuario (asociados a algún estilo y metáfora), qué transformaciones sucederán, cómo serán sincronizados los objetos de interface como audio y video, entre otros asuntos.

En [Rossi et al 95] se encuentran desarrollados los modelos para especificar el comportamiento estático de las interfaces basándose en el mecanismo de ADV (*Abstract Data View*). Se construyen diagramas de configuración para modelar las relaciones estáticas entre ADV, los eventos externos iniciados por el usuario y los objetos de interface que causan navegación. Para mostrar la dinámica de la aplicación se especifica para cada ADV su correspondiente carta-ADV (ver fig. 4.11).

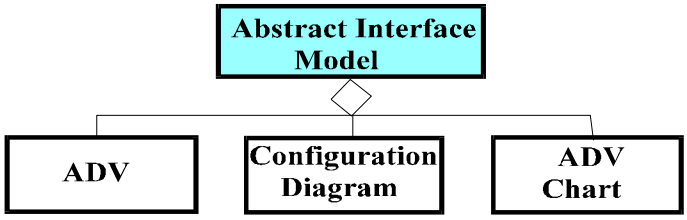
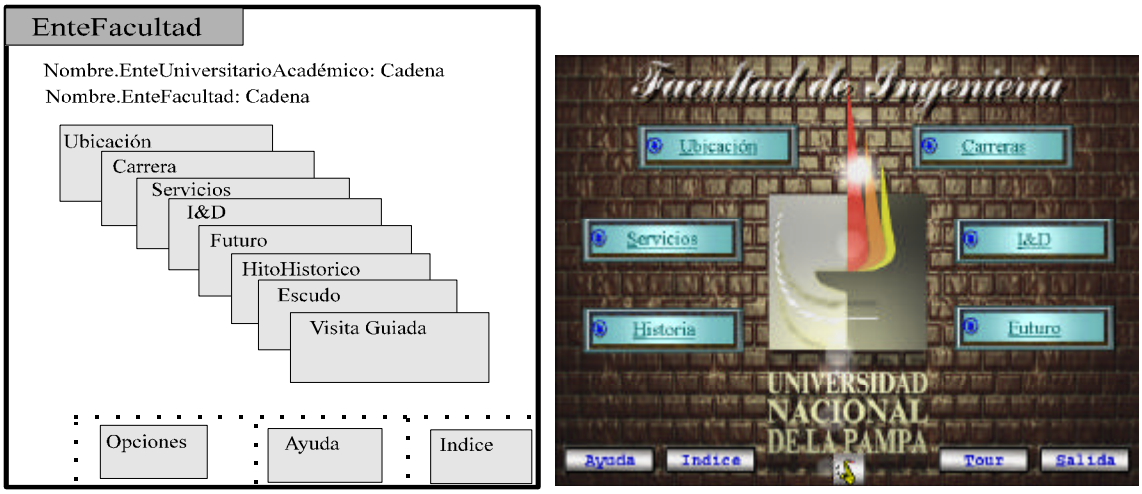


Fig. 4.11 *Diagrama de los constructores para el modelado de Interfaces Abstractas*

En la fig. 4.12a se especifica mediante una tarjeta de ADV la clase *"EnteFacultad"* y de aquí se mapea directamente a los objetos de implementación realizados en Multimedia Toolbook™ (fig. 4.12b). El ADV *"EnteFacultad"* está a su vez compuesto de varios ADVs entre los que se encuentran *"Ubicación"*, "*Carrera*", "*Indice*", etc.

En la tarjeta se puede apreciar recuadros punteados: en el caso de "*Ayuda*", "*Indice*" y "*Opciones*" involucran un comportamiento AND lo que significa que estos atributos

permanecerán visibles o percibibles por el usuario en los distintos nodos. En cambio si se selecciona el ADV "*Carrera*" los demás atributos como "*Ubicación*", "*Indice*", etc. no permanecerán visibles en el nuevo nodo (comportamiento XOR u o exclusivo). El comportamiento de un pop-up en tanto que no altere el contexto del nodo subyacente es un estado descripto por un AND.



a) b)

Fig. 4.12 a) *Tarjeta de especificación mediante ADV de la clase navegacional "EnteFacultad"*; **b)** *Un nodo de la aplicación de autoría "Facultad de Ingeniería"*

4.7. El modelado Físico

El proceso de *modelado físico* (tarea E de la fig. 4.1) se logra mediante el uso sistemático de prototipación flexible [Olsina 97b] la cual la consideramos una estrategia de desarrollo de artefactos de software. Esta estrategia participativa esencialmente promueve el ciclo de aprendizaje, descubrimiento, construcción, demostración y validación basada en la retroalimentación experimental entre usuarios y desarrolladores (tareas E, F, G).

Entendemos por *prototipo de software* al modelo físico implementado sobre computadoras el cual es una construcción parcial de todo el sistema o de componentes del mismo y permite la demostración del modelo en un lugar común de trabajo con el fin de descubrir, evaluar y refinar requerimientos funcionales y no-funcionales [IEEE 93]. El prototipo puede ser descartado o evolucionado.

Una premisa importante (tal vez no siempre válida) es que los prototipos constituyen un medio de comunicación mejor que muchas preespecificaciones en papel, y como dice la máxima “un cuadro puede valer más que mil palabras, pero un prototipo puede valer más que mil cuadros”.

En cuanto a *prototipación*, como indicamos anteriormente, la consideramos una estrategia de desarrollo caracterizada por un buen número de iteraciones y concurrencia con otras actividades, por un alto grado de participación de los usuarios, un uso extensivo de prototipos, y técnicas y herramientas avanzadas de producción.

Nuestra propuesta clasifica jerárquicamente a la estrategia de prototipación en tres clases concretas: *prototipación rápida-funcional* (PRF), *prototipación evolutiva* (PE), y *prototipación flexible orientada a objetos* (PFOO) que hereda el comportamiento de las dos anteriores. En la figura 4.13 se puede apreciar el diagrama de los principales módulos para la estrategia.

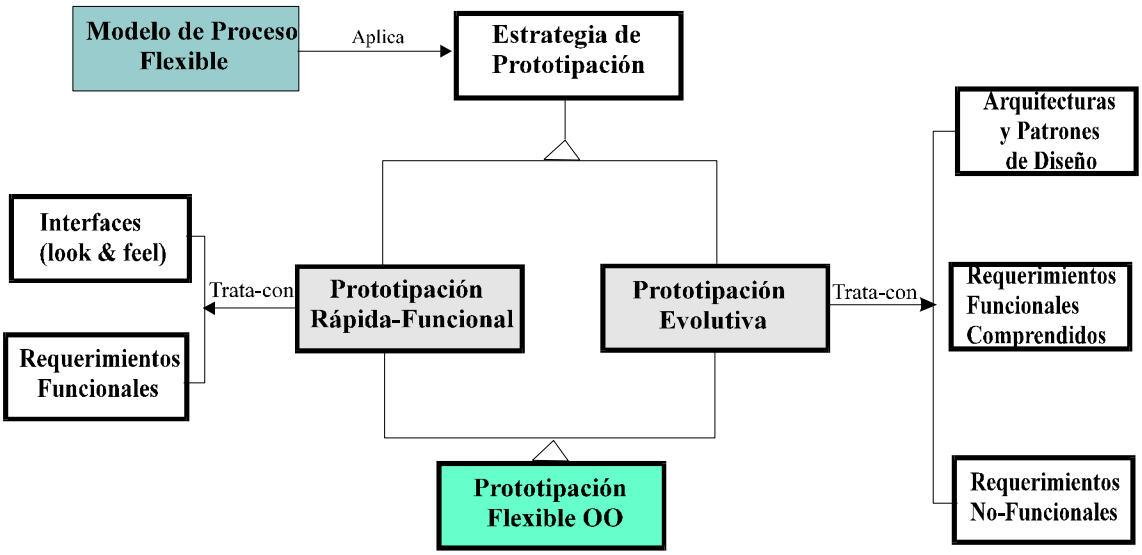


Fig. 4.13 *Diagrama de los principales responsabilidades de la estrategia de prototipación flexible.*

Algunas características y atributos del prototipo rápido-funcional y de la estrategia son*:*

- El prototipo rápido-funcional (PRF) se construye lo más rápido posible sacrificando la completitud aunque no la correctitud y la estética de la presentación; se debe considerar si será descartado o podrá reusarse y/o extenderse.

- El ciclo de desarrollo es bastante iterativo.
- Permite evaluar requerimientos funcionales entendidos por el desarrollador pero que necesitan ser validados experimentalmente; y permite capturar dinámicamente requerimientos pobremente entendidos (requerimientos funcionales críticos o no-críticos [Davis 92])
- Permite descubrir aspectos estéticos y de interacción de interfaces y patrones de navegación.

- Permite generar entradas a especificaciones de los modelos de requerimientos, conceptual, navegacional y de interface abstracta (atributos, controles, clases y contextos navegacionales, ADV, etc.). También genera entradas a tareas de documentación y de aseguramiento de calidad.

Algunas características y atributos del prototipo y de la estrategia de prototipación evolutiva son, a saber:

- Los atributos de completitud, correctitud y calidad del prototipo desarrollado son muy importantes. Es menos rápido que el PRF aunque planificado para su evolución hacia el componente o producto final.
- La entrada al proceso puede ser la evolución de características prototipadas y validadas por medio de un PRF, o de componentes de un software de la fase operativa.
- La estrategia es menos iterativa que la estrategia rápida-funcional.
- Permite construir sobre bases firmes los requerimientos críticos comprendidos; descubrir aspectos de arquitectura de diseño y verificar requerimientos no-funcionales (performance, etc.)
- Genera entradas a especificaciones del modelo conceptual, navegacional, de interface y testeo. También genera entradas a tareas de documentación y de aseguramiento de calidad.

En cuanto a la prototipación rápida-funcional, podemos a su vez clasificarla en prototipación vertical y horizontal.

La *prototipación horizontal* implica la reducción del nivel de funcionalidad del prototipo, concentrándose en cambio en aspectos estéticos y estructurales de la interface hombre-máquina. Permite obtener un rápida apariencia (look & feel) de la interface tan importantes en aplicaciones de autoría y en Internet.

La *prototipación vertical* permite atacar un problema disminuyendo la cantidad de características a considerar de la aplicación a cambio de obtener un artefacto (que es una parte reducida de la misma) con su funcionalidad bien entendida e implementada.

Desde un punto de vista práctico, la estrategia de prototipación rápida-funcional puede ser un híbrido de ambas estrategias.

La *prototipación flexible OO* hereda las características previamente definidas, y es flexible en tres sentidos. Primero porque dependiendo del problema a resolver ya se cuenta con la estrategia de PRF o con la estrategia de PE, o con ambas a la vez. En el mejor de los casos se aplica PE para construir y evolucionar los requerimientos mejor comprendidos y consensuados en tanto que los aspectos pobremente entendidos o que requieren validación con el usuario se le aplica una estrategia rápida-funcional. Así la prototipación flexible ofrece una base de componentes de software que son los cimientos sobre el cual evoluciona el sistema, usando en lo posible el ambiente y lenguaje de implementación de la aplicación final. Segundo, la estrategia es flexible por la retroalimentación y el equilibrio que se establece por el uso sistemático de modelado lógico y de modelado físico, basados en principios, técnicas y herramientas orientadas a objetos. Por último, la estrategia es flexible por su caracter evolutivo, el cual favorece la extensión y evolución de los artefactos.

Dentro de las contribuciones al mundo de la prototipación y por citar las de algunos autores como Boehm y otros, [Boehm 88, Connell et al 95, Davis 92, Nanard et al 95, Rudd et al 94] no tratan el concepto de prototipación flexible como una estrategia fundamental de soporte al ciclo de desarrollo enfocado a alimentar a la especificación de requerimientos, al diseño, a la construcción y validación de aplicaciones de la manera en que proponemos. Sobre todo en el campo de desarrollos de hipermedia que es bastante reciente, no es conveniente la aplicación de modelos de ciclo de vida tradicionales, por las razones expuestas en la sección 2.2.

4.8. Relaciones entre modelos

Como se ha indicado previamente, existe una clara división de preocupaciones entre el modelado de requerimientos, el modelado conceptual, el modelado navegacional, el modelado de interfaces abstractas y el modelado físico .

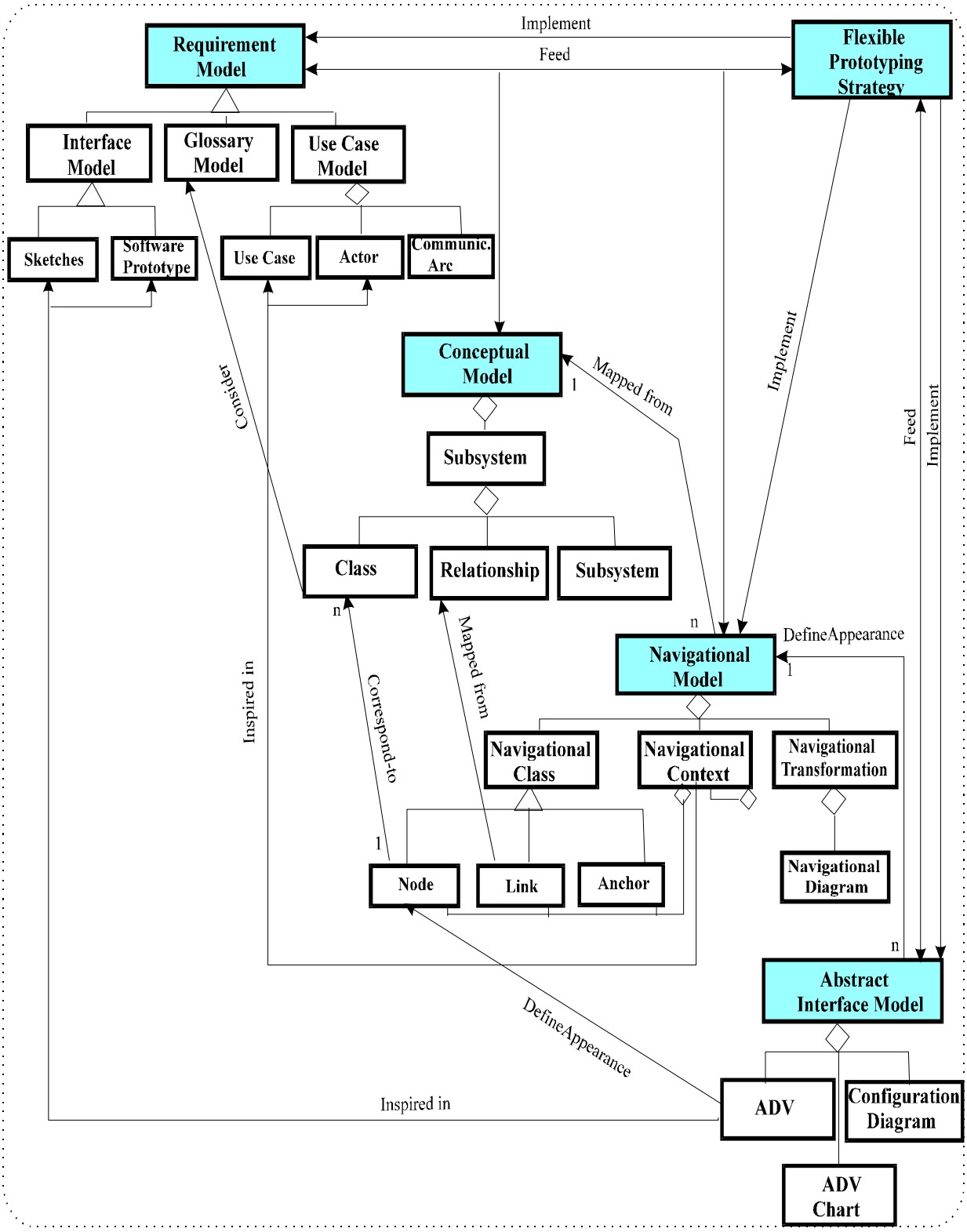


Fig. 4.14 *Diagrama que relaciona los modelos de requerimientos, conceptual, navegacional y de interfaces abstractas con la estrategia de prototipación flexible.*

Por ejemplo, la principal diferencia entre clases del modelo conceptual, y clases de caso de uso del modelo de requerimientos, reside en que las primeras se comunican con otras clases dentro del mismo sistema, mientras que las segundas no se pueden comunicar con otras clases caso de uso sino solamente con actores, fuera del sistema. Esto nos indica que el modelo de requerimientos se focaliza en la visión externa del sistema en tanto que el modelo conceptual pone énfasis en la visión interna y general del mismo. El hecho de que requerimientos sea una especificación general le confiere atributos de reusabilidad y poder de comunicación.

A su vez, el modelo navegacional es una visión mas específica que el modelo conceptual. Como se observa en la fig. 4.14, un nodo del modelo navegacional representa una ventana lógica de las clases definidas en el modelo conceptual y puede agrupar atributos de una o más clases (enlace "*correspond to*").

Los enlaces se traducen en relaciones y lo que es importante destacar es que se pueden diseñar "n" vistas navegacionales a partir del mismo modelo conceptual (enlace "*mapped from*"). Esto nos permite construir diferentes aplicaciones de hipermedia definidas como distintas visiones a partir del mismo esquema conceptual.

En el caso en que en un proyecto dado se fuera a construir un único perfil de usuario a partir del modelo conceptual es posible que se pase a especificar directamente el modelo navegacional, por restricciones de tiempo y costo, obviando la especificación conceptual. No obstante el modelo de requerimientos sigue siendo útil no sólo por lo explicado previamente sino porque también sirve como mecanismo de control de los distintos modelos. Se debiera validar/verificar un modelo (de diseño, de implementación) a partir de los requerimientos.

Como es sabido, la construcción de modelos está inserto en un proceso de desarrollo de artefactos de software, que denominamos modelo de proceso flexible. Como se discutió previamente, el modelado de requerimientos emplea modelos lógicos y físicos. El mismo es un buen mecanismo conceptual para asistir tanto en la creación de prototipos como para definir tareas y, en definitiva, el perfil del usuario. Podríamos afirmar que el perfil del usuario queda definido por el conjunto de roles encontrados en la elicitación de requerimientos. Cada caso de uso posee un objetivo, es decir, lo que el usuario desea explorar o encontrar. El caso de uso queda instanciado por una secuencia de acciones que el usuario realiza a partir de una acción inicial hasta llegar a una acción final. Por lo tanto, explorar distintos caminos para recorrer a un espacio de información relacionado (por ejemplo "Carreras"), nos puede ser de gran ayuda en la especificación de contextos navegacionales. Cada paso de la secuencia de uso, puede

corresponder a una relación o a un enlace, cada punto de partida puede corresponder a un anchor. El conjunto de acciones asociadas a los nodos origen y destino respectivamente, el modo de recorrerlos, las transformaciones producidas en cada paso, puede conformar los elementos que definan a un contexto de navegación.

Para finalizar con esta sección, vemos en la fig. 4.15 un diagrama de responsabilidades generales del rol jugado por el modulo de prototipación flexible con otros módulos intervinientes. (Observe los diagramas de las figuras 3.6 y 4.14)

La estrategia de prototipación flexible usa "Recursos Tecnológicos", los que automatizan parte del MPF, y los "Recursos Humanos" controlan, ejecutan y participan en las distintas fases y actividades del proceso de desarrollo de hipermedia. Además, el "Modelo del Plan del Proyecto" permite direccionar las actividades de la estrategia de "prototipación flexible", la que a su vez alimenta al "Plan", a los "Modelos Lógicos", al "Modelo de Requerimientos" y a los "Criterios Cognitivos y Estéticos" (en el ciclo de retroalimentación experimental desarrollador-usuario). Estos a su vez producen y reusan "Artefactos de Software". Asimismo la estrategia de "PFOO" implementa a los "Modelos" tal como se aprecia con mayor detalle en la figura 4.14

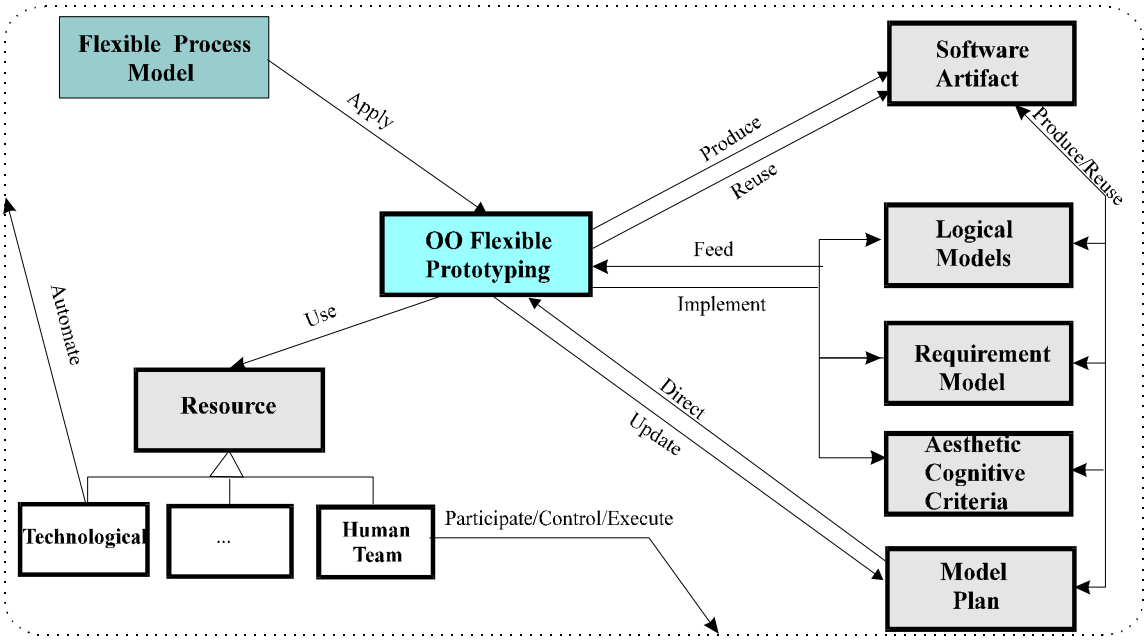


Fig. 4.15 *Diagrama general que relaciona la estrategia de prototipación flexible con algunos modelos lógicos y otras clases intervinientes.*

4.9 Algunos comentarios sobre la Perspectiva de Comportamiento (en la fase de desarrollo)

Como hemos indicado previamente, desde el punto de vista de la perspectiva de comportamiento, el modelo de proceso flexible ejecutado en la segunda fase es una combinación de estrategias *iterativa, concurrente, incremental y oportunística* guiadas por la prototipación flexible.

El carácter *iterativo* es por definición la clave del ciclo de la prototipación flexible: iterar significa que ciertas actividades se realizarán mas de una vez con el fin de mejorar a los modelos. Por cada iteración (del ciclo **E-F-G** -ver figura 4.1) el modelo físico se demuestra al usuario con el objeto de descubrir requerimientos adicionales, definir aspectos de arquitectura, validar la correctitud y la usabilidad del prototipo.

Esta estrategia permite un desarrollo *concurrente* entre los modelos lógicos y los modelos físicos durante la iteración: en cada ciclo se aumenta el prototipo y se actualizan las especificaciones. Además puede existir concurrencia, en un instante dado, entre la etapa de modelado navegacional y la etapa de modelado de interface abstracta, asimismo puede haber concurrencia entre aspectos de un prototipo rápido-funcional y aspectos de un prototipo evolutivo. Así, la concurrencia es una consecuencia de la no secuencialidad entre ciertas tareas, de una estrategia de particionamiento de un problema en subproblemas con cierto nivel de independencia y del enfoque evolutivo del modelo de proceso flexible.

Decimos que el proceso en la fase de desarrollo es *incremental*. Esto significa que habrá genéricamente, por cada iteración, incrementos del modelo conceptual, del modelo navegacional, del modelo de interface abstracta y de los modelos físicos. Por cada ciclo el prototipo se refina e incrementa. Luego de “n” iteraciones a lo largo del proyecto todos los requerimientos tenderán a estar completos y los artefactos de software entrarán en la fase operativa.

El carácter *oportunístico* del MPF en la fase de desarrollo consiste en que los analistas, diseñadores y programadores, en tanto se encuentran abocados a las tareas de prototipación y especificación, siguen un orden no siempre dictado por formalismos ni reglas sino mas bien por un proceso mental creativo, como acertadamente lo observan los autores en [Nanard et al 95].

Es común que un desarrollador de hipermedia pase de la prototipción de una interface de un

nodo, a la especificación de un nuevo contexto navegacional recién descubierto (oportunísticamente) y comience a bosquejarlo, para luego continuar con el PRF, y así siguiendo.

4.10 Criterios Cognitivos en el Diseño de Aplicaciones de Hipermedia

Así como es importante en el proceso de desarrollo de artefactos de hipermedia los modelos y la tecnología a emplear, igualmente importante son los mecanismos cognitivos usados por el usuario en la comprensión de hiperdocumentos.

El propósito u objetivo principal en la lectura de cualquier documento es la comprensión del mismo. Por lo tanto leer un hiperdocumento no es la excepción.

Collins [Collins 95] define el concepto cognitivo como el modo en que los usuarios procesan la información, usan la memoria, cómo perciben, piensan y actúan.

En ciencia cognitiva la comprensión de un documento por parte del usuario, se caracteriza por la construcción de un modelo mental el cual representa a los objetos y a sus relaciones estructurales y semánticas. La comprensibilidad de un documento se puede definir como el esfuerzo mental consumido por el usuario en el proceso de construcción del modelo. Por lo tanto, si queremos incrementar la legibilidad de un hiperdocumento debemos ayudar al lector en la construcción de su modelo mental, fortaleciendo aquellos factores que soportan al proceso y debilitando aquellos que la impiden.

En [Thüring et al 95] los autores exponen un conjunto de criterios y principios que el diseñador debe tener en cuenta de modo que ayuden al usuario en el proceso de construcción de su modelo mental, fortaleciendo a la *coherencia* entre los factores positivos y debilitando a la *demora cognitiva* como influencia negativa.

Estudios empíricos han demostrado que la capacidad de los lectores para comprender y recordar un texto depende del grado de coherencia del mismo. Investigaciones sico-linguísticas enfatizan la relación existente entre coherencia y procesamiento de información: un documento es coherente si el lector puede construir a partir del mismo un modelo mental que se corresponde a hechos y relaciones del posible mundo.

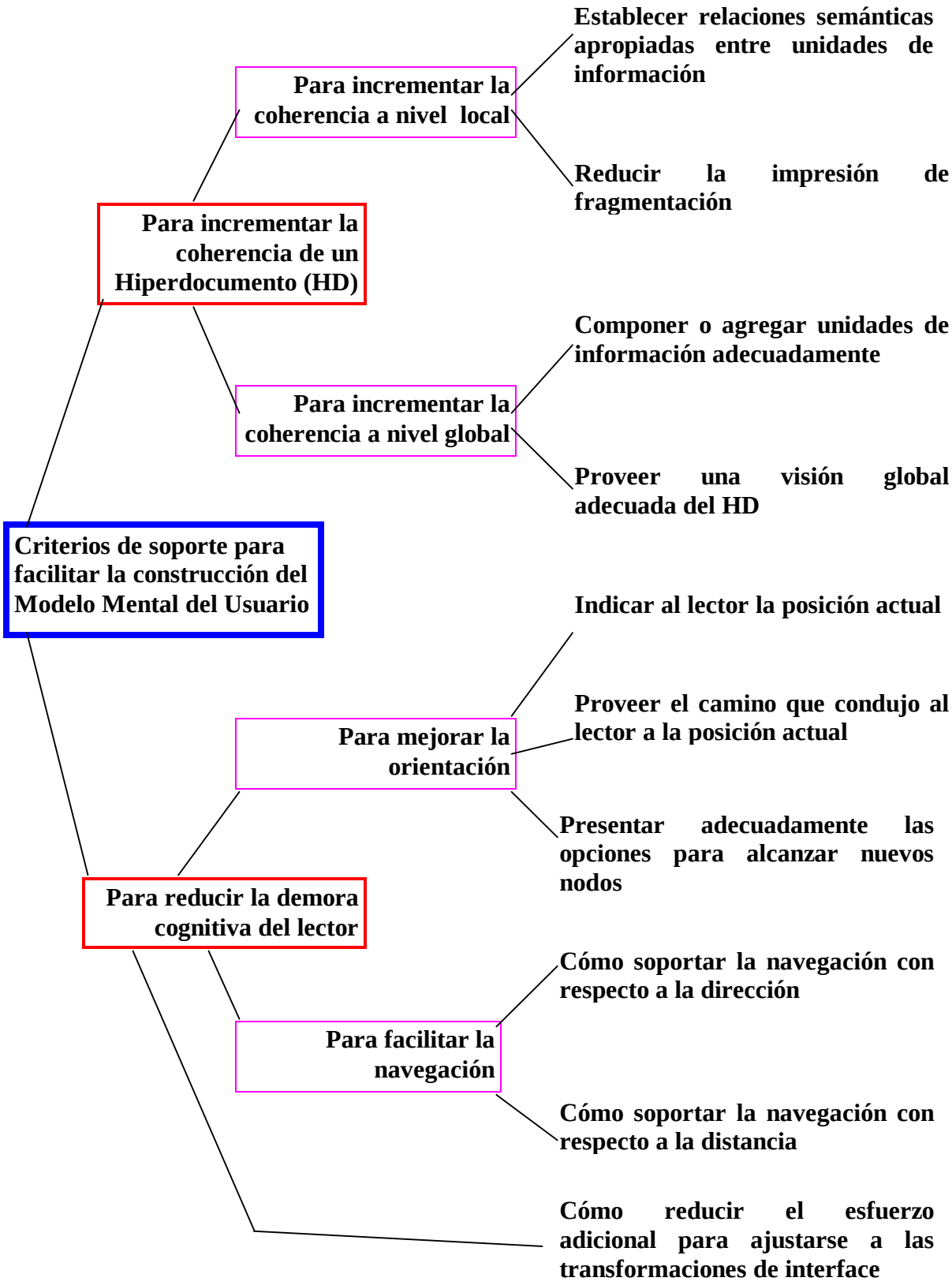


Figura 4.16: *Criterios Cognitivos en la creación de HiperDocumentos*

Para construir el modelo se debe tener en cuenta tanto la coherencia a nivel local (a nivel de nodo), como la coherencia a nivel global (en la red de nodos). Para incrementar la coherencia se debe limitar la fragmentación resultante de la segmentación de la información en nodos disjuntos que se muestran en ventanas separadas.

Una de las consecuencias de la fragmentación puede resultar en la falta de un contexto interpretativo y entonces parecer que el hiperdocumento es una agregación de piezas de información débilmente acopladas antes que un todo cohesivo.

En el siguiente esquema, tomado de Thüring et al (ver la figura 4.16), podemos considerar un conjunto de diez criterios a tener en cuenta en el desarrollo de aplicaciones de hipermedia.

A partir de estos criterios, los autores establecen ocho principios de diseño, que pueden ser empleados como base de actividades manuales de un agente o incorporados en herramientas de soporte a las aplicaciones de autoría.

Finalmente, podemos agregar que los contextos de navegación en el modelado navegacional (de OOHDM) son una respuesta a los tres criterios primeros y al menos al principio de componer y ordenar unidades en un alto nivel de abstracción. El uso sistemático de contextos de navegación tiende a minimizar el problema de la desorientación del usuario y reducir la demora cognitiva impuesta por la navegación en espacios complejos de información.

4.11 Otros asuntos

Los recursos tecnológicos son de gran importancia para el proceso de desarrollo de artefactos de hipermedia. Este ente está integrado por componentes de software y hardware y representa el soporte computacional al proyecto, las plataformas, sistemas operativos, herramientas y ambientes de soporte a procesos de software.

La arquitectura de un ambiente de software centrado en procesos debería ofrecer servicios de administración de objetos e interrelaciones, servicios de herramientas, servicios de trabajo colaborativo, y otros como administración de cambios, versionamiento, seguimiento de artefactos, seguridad, etc. como indicamos en la sección 3.1.

Para la realización del proyecto “Facultad de Ingeniería” no contamos con un ambiente de software centrado en procesos con las facilidades antes indicadas. Las investigaciones en este tipo de ambientes es inicial y sólo existen ambientes en estado experimental.

Por otra parte, sí existen disponibles, ya experimental o comercialmente, una vasta cantidad de herramientas y ambientes de autoría para dar soporte a varias actividades de un proyecto. Desde el punto de vista experimental, Lyardet et al [Lyardet et al 96] están construyendo una herramienta de automatización de varias actividades, denominada OOHDM-CASE, la cual favorecerá el desarrollo de aplicaciones de hipermedia con los constructores de dicha metodología y de la estrategia de prototipación flexible, esencialmente en las actividades de especificación, construcción y seguimiento.

Ampliamente conocidas son las herramientas y ambientes de autoría para generación de CD-ROMs y aplicaciones en Internet. En nuestro caso, una de las herramientas utilizadas en el proyecto (desarrollado durante el período 10/95 al 06/96), fue Multimedia Toolbook™ que satisfizo los requerimientos de prototipación (Actualmente estas herramientas permiten generar aplicaciones en el contexto de Internet). Esta herramienta (la utilizada en el proyecto) posee algunas características de orientación a objetos y tiene fortalezas como:

- Entorno de programación visual (rápida creación de nodos, atributos , controles de navegación, etc.). Este entorno es apropiado para el soporte a la estrategia de prototipación flexible.
- Ambiente interpretado (permite compilación) con la ventaja de ver y chequear las modificaciones rápidamente. Cuenta asimismo con un debugger potente.
- Permite importar fácilmente imágenes, animaciones en 3-D, videos ofreciendo un soporte práctico y fácil para incorporar multimedios.

y como debilidades:

- El modo restringido del mecanismo de herencia.
- No soporta muchas de las primitivas discutidas (creación de nuevas clases, soporte directo a contextos de navegación).
- No permite derivación de un modelo a otro
- No ofrece un modelo de seguimiento, ni de versionamiento, ni de configuración de cambios, etc.

En cuanto al perfil de usuario considerado en el proyecto se modeló al estudiante y se construyó esta vista del modelo conceptual (ver fig. 4.8).