Ingeniería de Hipermedia

Modelo de Proceso Flexible para el soporte sistemático al desarrollo de Aplicaciones de Hipermedia

Luis Antonio **OLSINA**

Tesis presentada al **Departamento de Informática de la UNLP** como parte de los requisitos para la obtención del título de
Master en Ingeniería de Software

Director de la Tesis: *Dr.* Gustavo Hector **ROSSI**

La Plata, Agosto de 1997; revisada en Febrero de 1998

Departamento de Informática
Facultad de Ciencias Exactas
Universidad Nacional de La Plata - Argentina

Agradecimientos

- ✓ Al Dr. Gustavo Rossi, por su material enriquecedor, por su guía y apoyo constante.

- ✓ A los revisores, porque con sus comentarios y observaciones me permitieron ajustar y enriquecer a la tesis.

- ✓ A los directivos de los Masters, por haberme posibilitado realizar los estudios de postgrado en mi país, allá en 1995, momentos en que eran inexistentes en estas áreas en las Universidades Nacionales.

- ✓ A la Facultad de Ingeniería, de la UNLPam, por el soporte económico facilitado que hicieron posible en buena parte este estudio

- ✓ A mi esposa Elena y a mis hijas, por soportar mi ausencia y acompañar mi esfuerzo con cariño y apoyo.

- ✓ A la gente del LIFIA por haberme facilitado las instalaciones del Laboratorio, y haber compartido momentos de trabajo y compañerismo.

- ✓ A mis colegas del grupo GIDIS, que me alentaron constantemente.

Indice

0. Resumen

El objetivo de esta tesis radica en definir y consolidar una estructura conceptual canónica de modelado de procesos de software, como así también personalizar un proceso de desarrollo que sirva para la comunicación y el mejoramiento de procesos y artefactos, en proyectos de hipermedia de mediana y gran envergadura.

En este trabajo, al modelo de proceso lo llamaremos Modelo de Proceso Flexible de Hipermedia (MPFH), que, instanciado en proyectos específicos, implica un uso sistemático de constructores basados en modelos lógicos (semi-formales) y físicos. Es de importancia considerar la participación de los usuarios en los distintos estadios de desarrollo como así también la adecuada elección de la tecnología a emplear.

Durante el desarrollo de la tesis presentaremos las principales fases, tareas y actividades del modelo de proceso flexible y un orden parcial en que las diferentes tareas se deben realizar. Además discutiremos métodos de proceso y sus constructores asociados (constructores para planificación, requerimientos, diseño conceptual, diseño navegacional y de interfaces abstractas, de prototipación, entre otros). Por otra parte, presentaremos las distintas perspectivas del modelo, a saber: funcional, de información, de métodos, de comportamiento y, en menor grado, comentaremos una perspectiva organizacional.

Mostraremos distintas alternativas, los aportes de la estructura conceptual y del proceso de desarrollo por medio de un caso de estudio. Finalmente, describiremos líneas abiertas de investigación para el futuro avance.

Abstract

The primary goals of this research are to define and consolidate a conceptual framework for process modeling and also, customize a hypermedia process model useful in communicating and improving processes and artifacts, mainly in a medium and large-scale hypermedia projects.

In this thesis, we propose an innovative and integrated software process model, called Hypermedia Flexible Process Model (HFPM). This strategy, when instanciated in specific projects, implies a systematic use of model-based constructors, both logical and physical ones. It is important to consider the user participation and the right choice of the employed technology.

During the development of the thesis we will show the main phases, tasks and activities of the three-phased Flexible Process Model and some partial order to take into account. We also will discuss different process constructors to build logical and physical models. In the other hand, we will present different perspectives, namely: functional, informational, methodological, behavioral and organizational perspectives.

We will discuss some contributions of the conceptual framework and we will use a case study to show some aspects of hypermedia development process. Finally, we will present concluding remarks and future works.

Keywords: Software Process Model, Hypermedia Project, Object-Oriented Technology, Artifact, Process, Agent, Process Constructor, Resource, Role.

1. Introducción

Desde comienzos de la década del 90, el advenimiento de los CD-ROM multimediales-interactivos y de la Web de Internet, han marcado un rápido crecimiento en los desarrollos de sistemas de hipermedia. Este crecimiento de aplicaciones, principalmente de mediana y gran envergadura, no ha sido acompañado por un proceso de desarrollo bien definido que favorezca el reuso y mejoramiento de los artefactos y procesos sino mas bien la estrategia utilizada ha sido una circunstancial, ad hoc, o como la cita Goldberg et al, la estrategia "just-do-it" [Goldberg et al 95] (centralizada en actividades de codificación, validación y corrección).

De manera que la Ingeniería de Software encara nuevos desafíos con el crecimiento de Hipermedia. Por ejemplo, surgen cuestiones de qué enfoques integrales utilizar de modo que las estrategias de proceso para la construcción de artefactos potencien el reuso, el mantenimiento, la dinámica del grupo, y, en definitiva, la calidad y la productividad dentro de las restricciones cada vez más exigentes de costos y tiempo. Podemos citar otros desafíos no triviales propios del reciente campo de hipermedia, a saber: cómo diseñar adecuadamente la estructura y el contenido de grandes espacios de información en hiperdocumentos; cómo permitir al usuario la navegación en dichos espacios de modo de facilitar la rápida comprensión del contenido y de la estructura (sin sentirse desorientado); qué aspectos cognitivos y estéticos deben primar en la construcción de los objetos perceptibles de las interfaces para facilitar la lectura de los hiperdocumentos e incentivar la atracción, etc.

Ya desde mediados de la década del 80 surgió el mayor interés, dentro de la Ingeniería de Software, de estudiar a los procesos con el fin de describirlos, comprenderlos, modelarlos y mejorarlos [Boehm 88, Humphrey 88, Osterweil 87]. Dichos estudios seminales han incluido aspectos de cómo describir lo que se observa y entiende acerca de los procesos; la creación de notaciones capaces de describir y especificar a los procesos; la necesidad de contar con herramientas y ambientes para controlar y mejorar a los procesos; cómo automatizar aspectos de los mismos, como producir artefactos de mejor calidad considerando costos, etc.

(Es importante tener presente en esta introducción, que un proyecto de software, cuando se instancia y ejecuta, involucra a un conjunto parcialmente ordenado de procesos que incluyen actividades técnicas, administrativas, cognitivas, con el fin de crear, evolucionar y mantener artefactos de software).

Por otro lado, para el proceso de desarrollo de artefactos de hipermedia podemos beneficiarnos en parte, ya de los modelos de proceso tradicionales [Berard 93, Boehm 88, Henderson et al 90, etc.], como de investigaciones sobre administración de proyectos en donde se pone énfasis en los modelos de objetos [Booch 96, Goldberg et al 95, Jacobson et al 92], como de las metodologías, principios y criterios de diseño discutidos en la literatura de Hipermedia [Isakowitz et al 95, Lange 94, Nanard et al 95, Rossi 96a, Schwabe et al 95b, Thüring et al 95].

Además, contribuciones al mundo del modelado físico han surgido varias y por citar algunas de [Boehm 88, Connell et al 95, Davis 92, Nanard et al 95] quienes en algunos casos aplican ciclos con prototipación y en pocos casos utilizan técnicas y métodos Orientados a Objetos [Connell 95, Nanard et al 95]. En general no tratan el concepto de la prototipación como una estrategia fundamental de soporte al ciclo de desarrollo que alimenta a las actividades de especificación, diseño, implementación y validación de artefactos de hipermedia según discutiremos.

Es importante apuntar que las metodologías y procesos de desarrollo actuales, en el área de hipermedia, se focalizan principalmente en fases de análisis y diseño, no teniendo en cuenta actividades que son fundamentales en un modelo de proceso integrado, basado en principios de Ingeniería de Software. En general no se consideran actividades como planificación, ejecución y control de proyectos, estrategias de aseguramiento de calidad, configuración de cambios, la conjunción de modelado lógico y físico, por citar algunas.

Por lo comentado antes, el marco o estructura conceptual que propondremos puede ayudar a comprender y modelizar la complejidad subyacente en todo proceso de desarrollo de software. Analizaremos los distintos objetos y sus relaciones principales. La definición de estos entes nos ayudará a definir las distintas perspectivas: funcional, informacional, de comportamiento, metodológica y organizacional.

Por otra parte describiremos, a un nivel de granularidad media, un proceso integral de desarrollo, el cual iremos ejemplificando a través de un caso de estudio. El mismo consiste en un modelo de proceso flexible [Olsina 97c] para soportar todo el ciclo de vida de artefactos de hipermedia en donde los modelos físicos se construyen con la estrategia propuesta de prototipación flexible [Olsina 97b], los modelos lógicos utilizados en tareas de diseño conceptual, navegacional y de interfaces abstractas se apoyan en la metodología de diseño de hipermedia orientada a objetos (OOHDM) [Rossi 96a, Schwabe 96]. También utilizamos modelos de requerimientos similar al modelo de casos de uso de Jacobson.

Asimismo en un proyecto de desarrollo específico se debe tener en cuenta el plan del proyecto, restricciones de tiempo y presupuesto, los recursos, el modelo mental y cognitivo de los usuarios en la etapa de diseño, la estrategia de reuso, un modelo de seguimiento, un modelo de configuración y de administración de cambios, entre otros asuntos.

Este enfoque de MPF, desde la perspectiva de comportamiento, permite actividades iterativas, concurrentes y particionamiento de un problema en subproblemas pudiéndose atacar incrementalmente.

Finalmente, una de las metas principales en el desarrollo de aplicaciones de hipermedia es producir artefactos de calidad, los que deben estar gobernados por un conjunto de características y atributos deseados [Fenton 91], utilizando para tal fin los procesos más óptimos y los recursos más apropiados. Debemos asegurar los mecanismos por medio de los cuales podemos construir artefactos de hipermedia que cumplimenten esas características.

1.1 Principales contribuciones

Las principales contribuciones de esta tesis para el campo de modelado de procesos en general y de proceso de desarrollo de artefactos de hipermedia en particular son, a saber:

- *consolidar una estructura conceptual canónica de objetos y relaciones esenciales.* Un principio básico que favorece al conocimiento público de carácter científico, consiste en identificar, representar y comunicar de un modo no ambiguo, un conjunto de conceptos generales y primitivos (conceptos como el de proceso, el de tarea, actividad, agente, artefacto, recurso, rol, constructores de proceso, entre otros)

- *definir de un modo integral, las fases y tareas generales de un proyecto de hipermedia abstracto, que pueda ser personalizado.* Esta clara división en fases, tareas y actividades favorece la visibilidad de un proyecto; además puede ayudar a la planificación, programación, ejecución y control de las mismas.

- *contribuir potencialmente a mejorar procesos y productos.* Esta declaración está motivada en que nuestra propuesta propende a un uso mas riguroso y sistemático de modelos y principios establecidos de la Ingeniería de Software (principios cognitivos, mecanismos de generalización/especialización y agregación, constructores arquitectónicos de alto nivel,

modelos de plan, de requerimientos, de navegación, de validación y evaluación; modelos físicos, estrategias de configuración de cambios, etc.)

- *redefinir y reusar los constructores de proceso* de modo que favorezcan al potencial empleo y equilibrio entre el modelado lógico (semi-formal), y el modelado físico en el proceso de desarrollo de un proyecto de hipermedia.

- *favorecer la estrategia participativa entre los distintos interesados en el proceso*. Para ello proponemos a la estrategia de prototipación flexible.

- *propender a la estandarización de procesos.* Dado que la mayoría de las tareas son comunes entre los distintos proyectos de hipermedia, una estructura estandarizada necesitará de menor esfuerzo para ser personalizada a las necesidades particulares de cada proyecto. Por otra parte esto tiene una implicación directa en el reuso de descripciones de proceso y de otros entes.

- *definir cuáles son las características y atributos que contribuyen a la calidad de artefactos, procesos y recursos.* Presentaremos enfoques de modelado de proceso que nos serán de utilidad para evaluar, analizar y controlar artefactos, procesos y recursos en el contexto de las metas establecidas y considerando los atributos deseables de un proyecto de hipermedia.

1.2 Estructura de la Tesis

En esta investigación presentamos un marco de referencia conceptual de clases y relaciones primitivas útil para todo modelo de proceso de software. Pensamos que la claridad de términos y conceptos son requerimientos básicos para posteriores trabajos.

Además discutiremos en particular el Modelo de Proceso Flexible para asistir en la creación, desarrollo y mantenimiento de artefactos de hipermedia. Analizaremos las fases, tareas y actividades y las distintas visiones de las cuales se puede enfocar al proceso de desarrollo. Reusaremos constructores de proceso ya establecidos y redefiniremos otros. Específicamente utilizaremos, a lo largo de la tesis, ejemplos del proyecto "*Facultad de Ingeniería"* [Olsina et al 95] y de otro más reciente, que corresponde a una aplicación educativa.

En el capítulo 2 discutiremos el estado del arte en procesos de desarrollo de hipermedia:

presentaremos sucintamente el problema que encara esta nueva disciplina de la Ingeniería de Software; luego discutiremos algunos modelos de proceso tradicionales; luego analizaremos metodologías establecidas de hipermedia en las que explícita o implícitamente definen procesos, actividades y artefactos sobre todo en las etapas de diseño de alto nivel, diseño detallado y en la etapa de construcción; finalmente describiremos en ese marco algunos de nuestros aportes a las investigaciones realizadas e introduciremos un ejemplo para contextualizar la discusión a lo largo del trabajo.

En el capítulo 3 esquematizaremos a un proyecto de hipermedia y describiremos los módulos esenciales en el contexto del modelo de proceso flexible, y, posteriormente, nos concentraremos en la definición del marco conceptual: analizaremos las distintas clases intervinientes y sus principales relaciones.

En un nivel de granularidad media describiremos, en el capítulo 4, un proceso integral de desarrollo. Especificaremos fases, tareas y actividades e iremos ejemplificando a través de modelos, constructores de proceso y estrategias específicas. Para los modelos físicos analizaremos principalmente la estrategia propuesta de prototipación flexible; dentro de los constructores lógicos emplearemos casos de usos, constructores de OOHDM, etc. Presentaremos las distintas perspectivas del MPF.

En el capítulo 5 argumentamos respecto de la necesidad de desarrollar artefactos de hipermedia de calidad, de utilizar los procesos óptimos y de seleccionar los recursos apropiados. Esto nos conduce a definir cuáles son los atributos y características que contribuyen a la calidad de artefactos, procesos y recursos. Además presentaremos enfoques de modelado de proceso que nos serán útiles para evaluar, analizar y controlar artefactos, procesos y recursos en el contexto de las metas establecidas de un proyecto de hipermedia.

Por último expondremos las conclusiones y los potenciales avances en distintas direcciones.

El lector encontrará a lo largo del trabajo referencias bibliográficas, que se encuentran al final, como así también un glosario de palabras y frases claves que son de importancia para la tesis. Además, cuando sea oportuno, se remitirá al lector a los distintos capítulos y secciones para facilitar la comprensión de este documento.

2. Estado del Arte en Procesos de Desarrollo de Hipermedia.

2.1 Presentación del Problema

No hace mucho se afirmaba que "los trabajos sobre modelado de proceso son recientes y la amplitud de la agenda de investigación aun está siendo formulada" [Curtis et al 92]. Si bien se conocen trabajos previos sobre modelos de proceso como indicamos en la introducción (y que ampliaremos en el siguiente punto), en la comunidad de hipermedia lo podemos considerar como un tema nuevo.

Para entrar en contexto utilizaremos algunas declaraciones de investigadores acerca de la necesidad de contar con modelos de proceso y lo que pueden aportar en la práctica de la creación, evolución y mantenimiento de artefactos de software en general.

"Cuando un grupo de personas trabajan cooperativamente sobre un proyecto común, necesitan de alguna manera coordinar su trabajo. En tanto las tareas son simples frecuentemente se pueden realizar de un modo informal, sin embargo cuando se incrementa el número de personas y la complejidad de las actividades, es preciso contar con mecanismos más formales..." "Se necesita un proceso de software bien definido para proveer a las organizaciones con un marco consistente para realizar y mejorar su trabajo" [Humphrey et al 89]

"Un modelo de proceso declara y establece un orden para realizar las actividades..." "Un modelo de proceso es una colección de máximas, estrategias, actividades, métodos y tareas, las cuales se organizan para alcanzar un conjunto de metas y objetivos" [Goldberg et al 95]

Queda claro que un modelo de proceso no es un conjunto de definiciones desordenadas de actividades sino que debe ayudarnos a comprender qué es importante para lograr esas metas y objetivos. Avanzando con algunos conceptos de los autores antes citados, Goldberg et al afirman que es importante definir un modelo de proceso porque ayuda a responder cinco preguntas críticas respecto de:

- planificación (qué tenemos que hacer para alcanzar nuestro objetivo?)

- autoridad (cómo podemos influenciar lo que está sucediendo para llegar a donde deseamos?)
- predicción (hacia dónde estamos yendo?)
- valoración (dónde estamos parados en el proceso de desarrollo y por qué?)
- seguimiento (cómo se alcanzó un determinado resultado?)

Además es oportuno decir que las investigaciones en modelado de procesos de software soportan un amplio rango de metas y objetivos. Curtis et al. enumeran cinco categorías fundamentales, a saber:

- *para facilitar la comprensión y comunicación entre los agentes involucrados en el proceso (agentes humanos: usuarios de proceso, desarrolladores de proceso, coordinadores, investigadores).* Se requiere estandarización de procesos, una representación comprensible y uniforme.

- *para soportar mejoramiento de procesos.* Se requiere una estrategia clara de definición, análisis y descripción de procesos.

- *para soportar la administración de procesos.* Se debe contar con un proceso de desarrollo de un proyecto, bien definido y flexible.

- *como guía automatizada en la ejecución de un proceso.* Esto es, definir guías, sugerencias en línea, material de referencia para facilitar a los agentes la ejecución de las tareas manuales o interactivas[1]. Esta guía automatizada puede estar embebida en un ambiente de desarrollo de software.

- *como soporte a la ejecución automatizada.* Es decir, de algunas porciones del proceso que se pueden controlar y ejecutar por medio de una interpretación mecánica de un modelo a partir de un servicio de descripción de procesos.

Desafortunadamente, la mayoría de las aplicaciones hipermedia se están desarrollando por medio de una estrategia ad-hoc. La falta de procesos bien definidos que cubran las distintas fases y actividades puede hacer crecer, en este nuevo campo, el fantasma de la crisis del software. Pese a que hipermedia se ha tornado un área de investigación extremadamente activa

[1] En la sección 3.1 o en el glosario se pueden encontrar las definiciones de estos conceptos

realmente hay muy poca literatura específica sobre el tema de modelos de proceso de hipermedia. Una observación interesante es analizar cuántos papers se han venido publicando en congresos de reconocimiento académico como Hypertext 97, Hypertext 96 y eventos anteriores u otros relacionados: el lector no encontrará material específico. No es el caso por ejemplo, con metodologías como OOHDM o RMM en las que se han publicado varios trabajos. En estas metodologías subyacentemente se definen actividades del proceso de desarrollo, principalmente en actividades de diseño, como luego veremos.

Sin embargo, en la comunidad científica se está comenzando a considerar la necesidad de contar con un enfoque amplio de ciclo de vida de desarrollo de hipermedia (en Hypertext 97 se dictó un tutorial sobre Ingeniería de Hipermedia, y se presentó un poster sobre un modelo de proceso flexible para la construcción de artefactos de hipermedia [Olsina 97a]). Principalmente se está viendo la necesidad de contar con un enfoque ingenieril; esto es, el empleo disciplinado, cuantificable y sistemático de principios de Ingeniería de Software para la creación, evolución y mantenimiento de artefactos de software de hipermedia.

La presente tesis pretende realizar un aporte ingenieril al proponer un modelo de proceso flexible que se adecue a la construcción de artefactos de hipermedia.

2.2 Trabajos Relacionados y Aportes

En este punto analizamos algunas características de procesos de ciclo de vida tradicionales que son públicamente conocidos en la literatura de Ingeniería de Software y comentaremos qué cosas serán de utilidad para nuestro modelo de proceso flexible. Luego, analizaremos metodologías recientemente establecidas para el proceso de hipermedia. Al menos tres de estas metodologías utilizan constructores centrados en modelos, y explícita o implícitamente definen actividades y artefactos sobre todo en las etapas de diseño y construcción.

Un buen número de modelos de proceso han sido propuestos, entre ellos el modelo de cascada [Royce 70]; el modelo en espiral [Boehm 88]; el modelo recursivo/paralelo [Berard 93]; el modelo de fontana [Henderson et al 90] y variaciones sobre los mismos. Por ejemplo una adaptación al modelo de cascada, es el realizado por el Departamento de Defensa de EU, conocido como el estándar DoD-STD-2167A [DoD 88] donde prima un estilo centrado en la documentación.

Royce fue el primero en acuñar la frase modelo de cascada (waterfall model) para representar

una secuencia bien definida de etapas, establecidas con criterios de Ingeniería de Software. Ofrece una visión de alto nivel del ciclo de vida de software. Define las siguientes etapas o tareas principales: análisis de requerimientos, especificación, diseño, implementación, testeo, operación y uso. Las mismas se representan como una cascada. La salida de una tarea es la entrada de la próxima. Como consecuencia es una estrategia simple para planificar y controlar proyectos: favorece a la administración de los mismos. La programación es una secuencia de tareas; la ejecución es lineal; con la culminación de una tarea se genera y aprueba un artefacto final (documento, código, etc.). Este modelo fue bien recibido en las esferas gubernamentales dado que se puede realizar un seguimiento de los contratos pactados y pagar en función de los distribuíbles [Goldberg et al 95].

Si bien el mayor mérito fue introducir una visión ingenieril en el desarrollo de software en contraposición a estrategias ad hoc (citadas como "code and fix", "just do it", etc.), el modelo falla por su extremada simplificación de la realidad. Con frecuencia su comportamiento secuencial es contraproducente: es reconocido el costo creciente en la corrección de errores generados en etapas tempranas del proyecto y detectados en las etapas tardías. Además el modelo no reconoce la naturaleza evolucionaria del software. Entre algunas desventajas informadas por varios autores (y que será útil que el lector tenga presente como puntos de comparación para un proceso de hipermedia), se encuentran:

- define una visión estática de los requerimientos
- toma demasiado tiempo para mostrar resultados al usuario
- demora la detección y corrección de errores hasta etapas tardías
- no promueve el desarrollo participativo ni la retroalimentación experimental entre desarrolladores y usuarios
- no promueve la estrategia de prototipación
- no establece una estrategia de reuso de artefactos

Una de las motivaciones que tuvo Boehm con su modelo en espiral fue introducir análisis de riesgo, carente en modelos como el de cascada. A diferencia de éste, las etapas se definen cíclicamente representadas por una espiral que evoluciona desde el centro de un sistema de ejes cartesianos. Las cuatro etapas están contenidas en sendos cuadrantes.

Es una estrategia en la que el foco se coloca en la identificación del problema y la clasificación de estos en distintos niveles de riesgo. Es una estrategia cuyo estilo está centrado en la detección de riesgos (risk-driven strategy), a diferencia del modelo de cascada que está

centrado en la documentación (document-driven strategy). Las etapas primitivas se pueden resumir en:

- identificar objetivos, alternativas y restricciones
- evaluar alternativas, identificar y resolver riesgos
- desarrollar y verificar los productos
- planificar la próxima fase

El modelo en espiral introduce un conjunto de ideas nuevas respecto del modelo de cascada, a saber: valoración y resolución de riesgos; una estrategia iterativa e incremental; una estrategia de prototipación para ciertas actividades; reconoce el caracter evolutivo para ciertas actividades.

Para que la estrategia recursiva/paralela sea efectiva [Berard 93], debe atacar proyectos en los que su propia naturaleza permita particionamiento. Consiste en ver a un sistema como compuesto de subsistemas lo más independientes posibles. De este modo a un problema se lo puede descomponer sistemáticamente en subproblemas. Este subproblema representa a un componente independiente, al que a su vez se le pude aplicar descomposición (la parte recursiva). Además la estrategia recursiva/paralela permite trabajar simultáneamente sobre un conjunto de componentes (la parte paralela). Así como se aplica la estrategia de descomposición la estrategia de composición es también factible.

El ciclo de vida para cada componente sigue un orden: analizar un poco, luego diseñar un poco, luego implementar otro poco y después testear. Tanto este modelo de proceso como el propuesto por Henderson et al, son más apropiados para proyectos que usan principios y tecnologías de orientación a objetos.

Es oportuno decir que para la construcción de artefactos de hipermedia no es adecuado aplicar una estrategia secuencial como la propuesta en el modelo de cascada por las desventajas antes expuestas (no favorece la estrategia participativa ni la retroalimentación experimental, no favorece la evolución de los artefactos, etc., etc.). Hablando en un sentido amplio, podemos afirmar que dada la naturaleza de las aplicaciones de hipermedia, con propiedades de interacción/navegación/interface, una mezcla planificada de estilos iterativo, incremental, concurrente y oportunístico, guiada por prototipación flexible es el más adecuado a este tipo de desarrollos, como discutiremos en el capítulo 4. (El lector debe tener presente que existen distintos tipos de proyectos [Goldberg et al 95]: “el primero de su tipo”, “una variación del

mismo", "generación de componentes reusables", etc. para los cuales habrá que adaptar las estrategias del modelo de proceso).

Algunas características de los modelos recursivo/paralelo y en espiral nos serán de utilidad; sin embargo este último aplica el concepto de detección, análisis y resolución de riesgos por cada iteración lo que introduce demoras afectando el ritmo del proyecto e incrementando potencialmente los costos (sobre todo para proyectos de pequeña o mediana envergadura). Los modelos de proceso discutidos y sus estrategias deberán ser adaptados al desarrollo de artefactos de hipermedia ya que hay modelos, actividades, artefactos, recursos y roles que pertenecen específicamente a este nuevo campo.

Por otra parte, en esta última década han sido publicados un buen número de criterios, modelos y principios que ponen énfasis en estructurar y producir aplicaciones de hipermedia [Conklin 87, Garzotto et al 91, Grønbaek et al 94, Nanard et al 91], y recientemente se han establecido metodologías de hipermedia, sobre todo enfocadas hacia actividades de diseño, y apoyadas en modelos y principios de Ingeniería de Software. Entre ellas podemos citar a HDM [Garzotto et al 91, Garzzoto et al 93], RMM [Isakowitz et al 95], OOHDM [Rossi 96a, Schwabe et al 95a , Schwabe et al 96]. Presentaremos algunas características de las mismas y sus aportes a una estrategia integral de modelo de proceso. Finalmente, consideraremos algunos comentarios valiosos realizados por Nanard et al [Nanard et al 95] respecto de características de los entornos de desarrollo de hipermedia y su rol en el proceso de diseño.

Uno de los inconvenientes al diseñar artefactos de hipermedia es cómo representar los resultados del proceso. Modelos físicos como prototipos son necesarios pero no suficientes porque pueden resultar en ambigüedades e inadecuados como base de comparación en análisis de diseños alternativos (de alto nivel). Por lo tanto es necesario contar también con modelos lógicos, formales o semiformales, que soporten a las actividades de diseño y documentación.

HDM (Hypermedia Design Method) es el primer enfoque basado en modelos para especificar decisiones en actividades de diseño. Sin embargo HDM es un modelo de diseño y no una metodología como RMM (Relationship Managment Methodology) u OOHDM; es decir, no especifica un conjunto de actividades, el ordenamiento y el comportamiento de los pasos en las cuales se crean y evolucionan los modelos.

HDM contiene unas pocas primitivas de modelado. Un esquema en HDM está compuesto por un conjunto de entes y enlaces. Los entes están constituidos por componentes y cada

componente puede ser observado bajo diferentes perspectivas (unidades). Los entes y componentes se conectan por medio de enlaces estructurales o de aplicación. En este enfoque se emplean estructuras de acceso como índices, rutas guiadas para ordenar y facilitar el acceso a la información. Es importante resaltar ideas subyacentes en el modelo de diseño de HDM (el cual ha dado origen a ampliaciones y ha servido como cimiento de otras metodologías):

- ofrece un esquema de navegación explícito al definir distintos tipos de enlaces y entes
- sus modelos son independientes de los entornos de implementación
- ofrece primitivas de composición y estructuración jerárquica
- incluye el concepto de perspectivas de un atributo

A diferencia de HDM, en RMM como en OOHDM se reconoce un proceso de desarrollo de hipermedia. Ambas metodologías definen subyacentemente un comportamiento incremental e iterativo para actividades de diseño y desarrollo.

La metodología RMM recibió influencia de HDM y su sucesor HDM2. Es una metodología estructurada en el modelo de datos (modelo de entidad-relación, E-R). Comprende un conjunto de etapas de diseño de alto nivel, de diseño detallado y de construcción. Se puede resumir en las siguientes etapas o tareas primitivas: 1) diseño de E-R, 2) diseño (particionamiento en slices) de entidades, 3) diseño navegacional, 4) diseño de protocolos de conversión 5) diseño de pantallas de interface de usuario 6) diseño de comportamiento de tiempo de ejecución, 7) construcción y testeo.

En el diseño de E-R se analizan las entidades relevantes y las relaciones en el dominio de la aplicación. Los autores afirman la importancia de explicitar los enlaces entre los distintos objetos porque serán los caminos principales por los que los usuarios navegarán los ítems individuales de información. En la siguiente tarea, propia de aplicaciones de hipermedia, involucra el particionamiento de una entidad en trozos cohesivos de información o "slices" significativos, para después organizarlos en una red de hipertexto. RMM hereda de HDM los conceptos de enlace estructural y asociativo. "Desde el punto de vista navegacional, es importante la diferencia entre estos dos tipos de conexiones. Cuando un usuario atraviesa un enlace asociativo, el contexto de la información cambia, por ejemplo, de una facultad a un curso. No obstante, cuando se atraviesa un enlace estructural, la información del contexto queda dentro de la misma entidad" [Isakowitz et al 95]. En la tercera etapa se diseñan los caminos que posibilitarán la navegación; se analizan las relaciones asociativas descubiertas en la primera etapa. Además RMM cuenta con seis primitivas de acceso navegacional (índices,

tours guiados, etc.). En el proceso de diseño de protocolos de conversión se producen reglas de traducción del modelo previo en objetos de la aplicación (considerando el entorno de autoría: HTML, Toolbook, Macromind Director, etc.). Durante la quinta etapa se analizan esquemas de pantallas para los objetos perceptibles por el usuario. Las etapas de construcción y testeo no varían demasiado de los proyectos tradicionales; excepto que en aplicaciones de hipermedia se deben tener en cuenta los test de los principales caminos de navegación, entre otros asuntos.

A diferencia de HDM, RMM provee un proceso de diseño paso a paso y enriquece significativamente a sus primitivas. También es el caso de OOHDM el cual define un proceso de desarrollo de cuatro pasos y que a seguir describiremos. Es oportuno decir que tanto OOHDM como RMM son metodologías centradas principalmente en el diseño y construcción, no tratando actividades de etapas tempranas del ciclo de vida como por ejemplo análisis de factibilidad, planificación de proyectos, estrategias de desarrollo participativas como prototipación, y otras actividades que en el punto 2.4 introduciremos y en el capítulo 4 discutiremos con mayor detalle. Sin embargo OOHDM es la metodología de diseño más avanzada hasta la actualidad, por la potencia de sus constructores Orientados a Objetos (OO), por la clara división de preocupaciones entre las etapas conceptual, de navegación, de interfaces abstractas e implementación, y, principalmente, por la sonada estrategia de reuso por medio de patrones de diseño [Rossi 96b, Rossi et al 97].

La metodología OOHDM consta de las siguientes etapas o tareas fundamentales: 1) modelado conceptual, 2) diseño navegacional, 3) diseño de las interfaces abstractas, 4) implementación.

El objetivo fundamental en la etapa de modelado conceptual consiste en analizar y definir la semántica del dominio de la aplicación. El producto de esta tarea son diagramas de clases y objetos construidos a partir de clases, objetos, atributos (posiblemente multitipados), relaciones y subsistemas. Los autores consideran importante explicitar las relaciones entre los distintas clases porque serán los potenciales enlaces para los esquemas de navegación. No es preocupación en esta etapa el análisis de los tipos de usuario y de las tareas realizadas por ellos.

En el diseño navegacional (segunda etapa) se deben considerar principalmente aspectos cognitivos, el perfil de usuario para cada vista a construir a partir del modelo conceptual, las tareas realizadas y, en definitiva, encontrar los distintos contextos navegacionales. Las primitivas son los nodos, enlaces, clases y contextos navegacionales. Un contexto navegacional se define como un conjunto de enlaces, nodos y otros contextos. Se construyen tres esquemas fundamentales en esta etapa: el esquema de clases navegacionales y los esquemas de contextos,

y, para especificar la dinámica (es decir, el conjunto de objetos navegables accesibles a cada momento) se construyen las transformaciones de navegación.

La metodología OOHDM diferencia el diseño navegacional del diseño de interfaces abstractas. Esto permite construir diferentes interfaces para un mismo modelo navegacional. Los aspectos a tener en cuenta en este paso radican en definir qué eventos intervendrán en el lenguaje de acción y qué objetos de interface percibirá el usuario (asociados a algún estilo y metáfora); qué transformaciones de interface y de objetos navegacionales sucederán; cómo serán sincronizados los objetos multimediales de interface, principalmente los que incorporan audio y video. Básicamente se utiliza el mecanismo de ADV (Abstract Data View) para modelar cada objeto perceptible y se define un tipo de ADV para cada clase navegacional; para especificar las relaciones estáticas entre ADV, eventos externos iniciados por el usuario y objetos de interface que causan navegación se construyen diagramas de configuración; y para mostrar la dinámica de la aplicación se especifica para cada ADV su correspondiente carta-ADV.

En estas tres etapas existe una transición suave entre modelos lógicos basados esencialmente en los principios de objetos que sirven de entrada a una cuarta, de implementación, en donde los objetos de navegación y de interface se mapean a los objetos de implementación. En esta etapa se selecciona el entorno de hardware y herramientas de autoría apropiadas.

Es importante destacar los aspectos evolutivos de la metodología OOHDM con respecto a las discutidas previamente:

- *ofrece mecanismos de abstracción más ricos* (los modelos construidos están basados en el paradigma OO el cual soporta mecanismos como clases, generalización/especialización, agregación y subsistemas).
- *establece una clara división de preocupaciones entre etapas, proveyendo un conjunto bien definido de primitivas.* (Los constructores de proceso, al estar basados en modelos, permiten encarar proyectos de hipermedia de mayor complejidad, favoreciendo atributos de Ingeniería de Software como son la reusabilidad y mantenibilidad).
- *ofrece un conjunto de patrones arquitectónicos de alto nivel, útiles en el proceso de desarrollo, para documentar y reusar experiencias de diseño.* (Entre algunos patrones documentados en la bibliografía antes citada se encuentran: visiones navegacionales, contextos navegacionales, transformaciones navegacionales, observador de navegación, información bajo demanda, referencia activa).
- *permite definir un modelo de seguimiento*. (Esto favorece esencialmente mecanismos de

evolución y mantenimiento de artefactos en proyectos de hipermedia de mediana y gran envergadura).

Finalmente, en [Nanard et al 95] los autores concluyen que los métodos y técnicas formales mejoran la consistencia de las especificaciones y proveen guías bien definidas en las fases pero que buena parte de las preocupaciones en el diseño hipermedial surgen de cuestiones cognitivas y estéticas. Proponen que un ambiente de diseño hipermedial no sólo debe soportar técnicas formales sino que también debe incorporar un proceso oportunístico e incremental basado en la retroalimentación experimental.

2.3 Motivaciones

El modelado de procesos es la ciencia que sirve para representar y soportar a los procesos y a las distintas interacciones que forman parte de un problema del mundo real. Una de las dificultades en modelado de procesos es la complejidad inherente de los procesos de software de la realidad. Sólo basta con citar alguna de sus variables para comprender la complejidad subyacente: por ejemplo costos, recursos, habilidades, restricciones, artefactos creados evolucionados o reusados, duración del proyecto, transformaciones de procesos y artefactos, estructuras organizacionales, constructores de proceso, etc. conforman algunos ingredientes básicos de todo proyecto.

Los procesos de software se pueden abstraer en un modelo de proceso, es decir, podemos obtener un común denominador lo suficientemente significativo de manera que sirva con fines de comunicación, comprensión, análisis, guía y reuso en diferentes proyectos.

Un modelo de proceso debe ayudar a responder preguntas como:

- Qué hacer
- Cómo y cuándo hacerlo
- Dónde y quiénes lo harán
- Qué dependencias existirán entre tareas y otros entes
- Qué estrategia de modelado de procesos se utilizará conforme a las metas y objetivos del proyecto

Ampliando los conceptos de Goldberg et al. definimos a un modelo de proceso de desarrollo de software como a una estrategia apropiada para abstraer, organizar, ejecutar y/o controlar

(mediante el uso sistemático de heurísticas, métodos, modelos, técnicas y herramientas), a las distintas fases, actividades, recursos y artefactos de un proyecto con el fin de alcanzar las metas y objetivos establecidos.

Abstrayéndonos de las metas y objetivos particulares de cada proyecto, afirmamos que todo proceso de software debe velar continuamente por tres fines esenciales: desarrollar los artefactos de calidad, emplear los procesos óptimos y aplicar los recursos apropiados. Desarrollar los productos de calidad significa producir artefactos los cuales deban contener un conjunto de características y atributos deseados. Utilizar los procesos óptimos significa seleccionar los procesos más efectivos de manera de usarlos consistentemente para obtener la aplicación que cumplimente tales características.

Para los agentes intervinientes en el ciclo de desarrollo, los atributos representan una serie de restricciones a cumplir por los productos que se están construyendo, los procesos que se están empleando, y los recursos que se están asignando. Para que la descripción del modelo de proceso se complete es preciso definir las características y atributos que representen unidades de medida concreta para observar, retroalimentar, evaluar, analizar, mejorar y predecir.

Lo anterior da pie para definir cuáles son los atributos que contribuyen a la calidad en un proceso de desarrollo. Por ahora sólo diremos (ya que será tratado con mayor profundidad en la capítulo 5) que un proceso de calidad debe ser: repetible, correcto, medible, relevante, debe permitir ciclos de retroalimentación y aprendizaje conjunto, debe ser flexible (adaptable a varias definiciones de proceso, útil para ser instanciado en varios proyectos, escalable). Algunas de estas características y atributos para procesos (artefactos y recursos) son propias o deben ser adaptadas para el campo de Hipermedia (por ejemplo navegabilidad, relevancia de enlaces, confiabilidad, usabilidad, etc.)

De manera que si bien muchas de las características y estrategias de los modelos de proceso comentados en la sección 2.2 nos serán de utilidad (como estrategias iterativas, incrementales, prototipación, análisis de riesgo, etc.), nuestra propuesta es definir un modelo de proceso flexible que se adecue totalmente al desarrollo de artefactos de hipermedia. Como observamos anteriormente hay modelos (por ejemplo de calidad), entes (como actividades, artefactos, recursos, roles, constructores de proceso) y atributos de estos entes que pertenecen específicamente a este reciente campo de investigación y desarrollo. Dada la naturaleza de las aplicaciones de hipermedia, con características cognitivas y estéticas, de interacción, navegación e interface, una mezcla planificada de estrategias iterativa, incremental, concurrente

y oportunística, que soporte prototipación y reuso es lo mas adecuado para este tipo de desarrollos. Esta mezcla de estrategias debe ser planificada y se ajustará dependiendo del tipo de proyecto y producto.

Además un modelo de proceso debe abstraer e integrar diferentes tipos de información de los procesos de software en perspectivas. Una perspectiva es un enfoque particular, una vista, para especificar y comunicar información del modelo. Para tratar con la complejidad inherente de los procesos nosotros proponemos perspectivas a partir del modelo conceptual del dominio de modelado de procesos en general. Curtis et al. agrupan la información en cuatro perspectivas con el fin de abstraer la complejidad subyacente en los procesos de software, y son, a saber: perspectiva funcional, perspectiva informacional, perspectiva de comportamiento y perspectiva organizacional. Nuestra taxonomía amplía la propuesta por Curtis et al. incorporando una perspectiva metodológica la cual es fundamental para representar a los diferentes constructores de proceso para realizar las tareas (y además puede ser integrado en ambientes de ingeniería de software centrado en procesos). En el capítulo tres y cuatro se estudian algunos de estos aspectos.

En cuanto a OOHDM y RMM comentamos que son metodologías centradas principalmente en actividades de diseño y construcción, no tratando actividades de etapas tempranas del ciclo de vida, ni estrategias participativas de desarrollo, ni actividades de mantenimiento. No obstante desde el punto de vista de la perspectiva metodológica nos serán de gran utilidad para definir a los constructores de proceso que se emplearán para especificar, personalizar y ejecutar las descripciones de procesos correspondientes a las tareas de diseño de alto nivel y diseño detallado.

Si bien se pueden especificar distintas descripciones de proceso a las que se les puede aplicar diferentes constructores, nosotros emplearemos en los ejemplos (capítulo 4) los constructores de OOHDM, por las razones expuestas al final de la sección 2.2. (Por ejemplo, para la tarea de modelado conceptual, una descripción se podría personalizar y luego ejecutar con los constructores de RMM, basados en los diagramas de E-R o con los de OOHDM, basados en diagramas de clases, objetos y relaciones).

Para ingeniería de requerimientos aplicamos modelos y constructores de proceso centrados en casos de uso y modelado de interfaces (sección 4.3).

Ultimamente, en la comunidad científica de hipermedia, se está comenzando a considerar la

necesidad de contar con un enfoque amplio de proceso de desarrollo. Principalmente se está observando que es preciso contar con un enfoque ingenieril; esto es, el uso disciplinado, cuantificable y sistemático de principios y características de la Ingeniería de Software para la creación, evolución, evaluación y control de artefactos (y otros entes) en proyectos de hipermedia.

2.4 Panorama de las principales fases y procesos del MPF

Introduciremos las principales fases y procesos del modelo de proceso propuesto, a un alto nivel de abstracción, y sin hacer hincapié en las diferentes perspectivas (aunque lo que sigue se enfoca principalmente en aspectos funcionales).

El modelo está comprendido en tres fases generales [Olsina 97c] como se ve en la fig. 2.1. En cuanto a una elipse representa a un subsistema proceso (la cual puede estar compuesta jerárquicamente de actividades), y las flechas representan dependencias funcionales de entrada y salida (por ejemplo un proceso depende de la culminación de un artefacto de otro proceso o a un proceso se le envía un mensaje de control). En los laterales de la figura se puede apreciar las principales fases de la vista.

La primera fase se denomina *fase de exploración* en la cual se elicitan conceptos y requerimientos iniciales, si fuera necesario se realiza un estudio de factibilidad y se planifica preliminarmente.

La segunda fase, *de modelado dinámico*, es la fase esencial de re-planificación, elicitación y especificación de requerimientos detallados, coordinación y control, análisis del dominio, diseño, construcción, validación, integración, empleo de criterios estéticos y cognitivos, aseguramiento de la calidad y documentación. Se hace uso intensivo de modelos lógicos y físicos.

La tercera fase, denominada *fase operativa* o de vida útil de los artefactos, consiste esencialmente en tareas de documentación, configuración de cambios, mantenimiento y evolución de los artefactos. En cuanto a la tarea de mantenimiento y al igual que en Ingeniería de Software tradicional podemos considerar tres tipos de mantenimiento: correctivo, adaptativo y perfectivo.

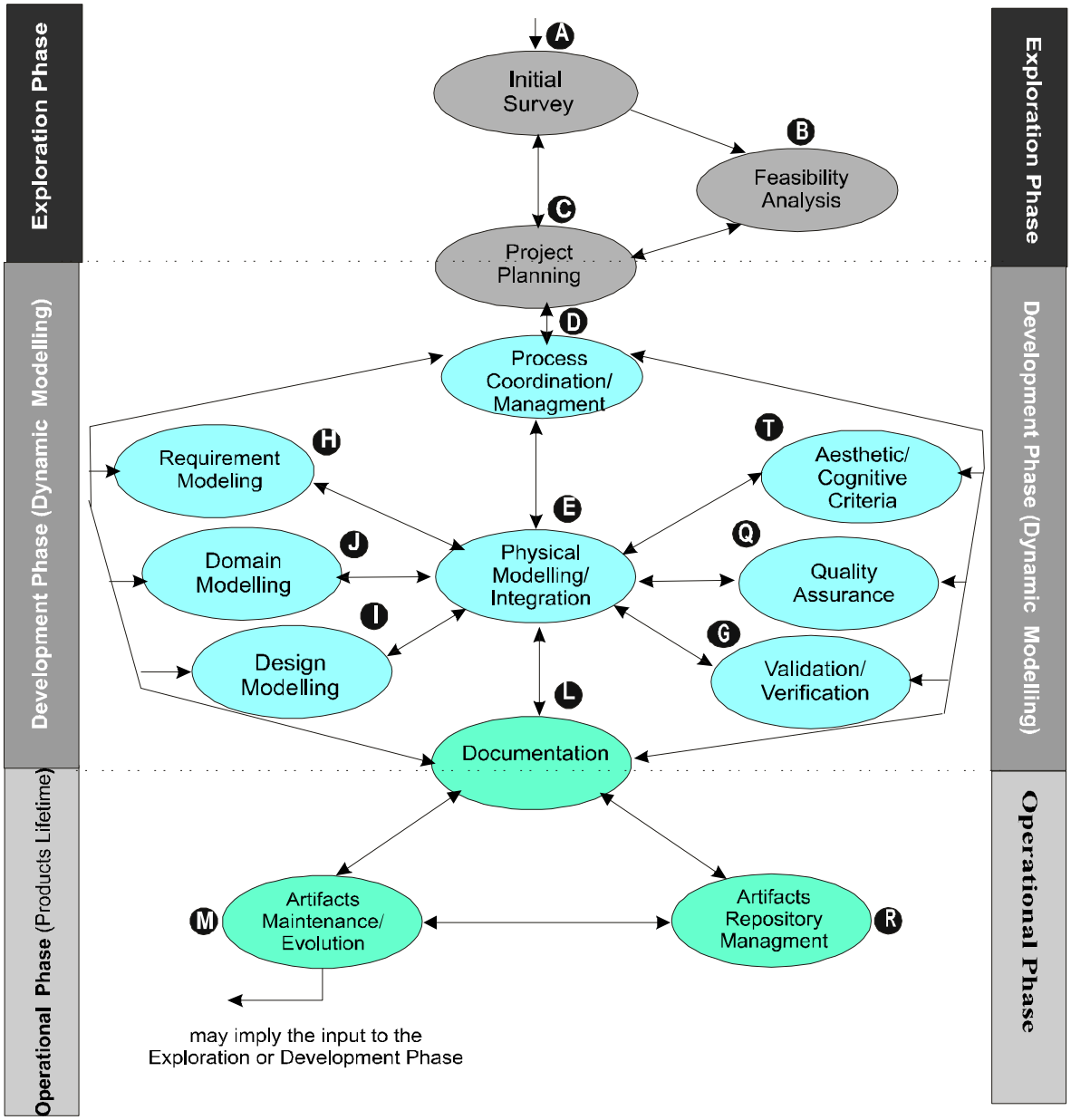


Fig. 2.1 Panorama de las fases y procesos principales del modelo de proceso flexible

Antes de comenzar el proceso de modelado detallado del dominio y de diseño arquitectónico se debe conceptualizar sobre el problema e identificar requerimientos iniciales. La entrada al proceso **A** puede ser en el mejor de los casos la extensión de un sistema en operación con

requerimientos documentados (de la fase operativa) o puede implicar la construcción de una nueva aplicación o componente en donde se necesita identificar el vocabulario del dominio, analizar a los tipos de usuarios y sus tareas y estudiar alternativas (dado el carácter de nuevo que tiene la disciplina de hipermedia es difícil encontrar componentes o aplicaciones documentadas que sirvan de entrada a los requerimientos iniciales). Aquí usamos el modelo de requerimientos para elicitar el dominio del problema. Se puede determinar inicialmente un conjunto de abstracciones claves y una secuencia de acciones que defina los modos específicos en que distintos actores interactuan con el sistema y establecer casos de uso [Jacobson et al 92]. La salida de esta tarea es una especificación inicial de requerimientos (posiblemente expresados en lenguaje natural o en algún otro tipo de descripción de procesos).

A partir de los requerimientos iniciales se puede extraer posteriormente información de cómo los usuarios realizan sus tareas y reconocer a objetos, transformaciones, nodos, contextos de navegación u otras primitivas arquitectónicas.

Es necesario confeccionar un modelo del plan inicial del proyecto, por medio de la tarea **C**. Básicamente contendrá, a partir de los requerimientos iniciales y del análisis de factibilidad (tarea **B**): una descripción del objetivo del sistema de hipermedia a construir o extender, el alcance previsible del producto final, los usuarios involucrados en el proceso y sus respectivas responsabilidades, las estrategias del modelo de proceso a utilizar, la elección del ambiente de desarrollo y de la herramientas, la posibilidad de establecer métricas en el contexto de un modelo de calidad. Como se puede ver esta actividad abarca las dos fases, por lo que el documento debe incluir la fecha y el número de revisión que sean de utilidad para el coordinador del proyecto en posteriores iteraciones.

Como se puede observar la segunda fase está conformada por siete componentes mutuamente necesarios para el proceso de desarrollo de hipermedia: el proceso **D** que corresponde al proceso coordinador y controlador de otros procesos y actividades (asignado por ejemplo a un agente humano que cumpla el rol de administrador del proyecto); el componente integrado por los procesos de alto nivel **H-J-I** que corresponden a procesos principalmente de modelado lógico y especificación independientes del entorno de implementación; el componente de tareas **E-G** guiado por la estrategia de prototipación y que corresponden a un esquema de iteración/retroalimentación experimental desarrollador-usuario para tareas de diseño, construcción y validación. Como se aprecia en la figura la estrategia de prototipación alimenta asimismo a las actividades del modelado lógico. El proceso **L** que comprende la documentación y se comunica con el repositorio de administración de artefactos (proceso **R**);

el proceso **Q** que corresponde a actividades de aseguramiento de la calidad de los procesos y, esencialmente, de los productos; el proceso **C** de replanificación del proyecto, y el proceso **T** para el empleo de criterios cognitivos (tratado en la sección 4.10) y estéticos en el desarrollo de aplicaciones de hipermedia.

El comportamiento del modelo de proceso, principalmente el ejecutado en la fase de desarrollo es una combinación planificada de estrategias *iterativa, concurrente, incremental y oportunística, guiada por prototipación flexible* (referirse a la sección 4.9).

2.4.1 Ejemplo a utilizar

Dentro de los proyectos realizados se encuentra un trabajo de autoría denominado "*Facultad de Ingeniería*" que utilizaremos para ejemplificar distintos aspectos del proceso de desarrollo.

El proyecto se construyó empleando el modelo de proceso introducido y el dominio de la aplicación representa una vista del esquema conceptual de un Sistema de Información Académico (fig 4.9). El perfil del usuario considerado es el del estudiante. El objetivo de la aplicación consiste en que potenciales ingresantes de distintas regiones conozcan facetas de la Facultad y les ayude a seleccionar los centros académicos al que concurrirán.

Desde el punto de vista de autoría la misma utiliza distintos recursos de multimedia (videos, animaciones, textos, sonidos, imágenes). En la fig. 2.2 apreciamos dos secuencias de un video de presentación, integrado en un ambiente de desarrollo Multimedia Toolbook ™.

Desde el punto de modelado lógico la aplicación está diseñada en varios contextos. Un contexto de navegación está compuesto esencialmente por clases navegacionales como nodos, enlaces, anchors y otros contextos. Representa una unidad o espacio de información semánticamente cohesiva y navegable de un modo no necesariamente secuencial. En la fig. 2.3a se aprecia el nodo principal (implementado) en donde se seleccionan los distintos contextos temáticos de la clase *"EnteFacultad"*. Podemos apreciar los anchors de los contextos "*Ubicación*", "*Carreras"*, "*Futuro*", etc. En la fig 2.3b mostramos la pantalla del primer nodo destino del anchor *"Carreras"*

Fig. 2.2 *Dos secuencias de la presentación inicial de la aplicación "Facultad de Ingeniería".*

a) b)

Fig. 2.3 a) *Menú principal o nodo superior en la jerarquía de la aplicación hipermedial;* **b)** *Primer nodo del contexto "Carreras"*

En la figura 2.3a podemos además apreciar dos estructuras de acceso como "*Índice*" y un "*Tour*" guiado. El índice tiene el mismo contexto temático que el mostrado en el menú pero por cada tema agrega en orden alfabético los subtemas correspondientes: así el usuario puede acceder de un modo directo a la información de su interés.

En cuanto al tour o visita guiada, está en un contexto navegacional que considera exclusivamente al estudiante dentro del perfil de usuario seleccionado. Se navega por la hiperbase teniendo en cuenta esencialmente información de "*Ubicación*", "*Carreras*" que además incluye información de requisitos de ingreso, "*Servicios*" al estudiante como becas, viviendas, etc.; y cierta información de la perspectiva futura de la Facultad.

Actualmente estamos trabajando en un proyecto de hipermedia cuyo fin es una aplicación educativa para el soporte a la enseñanza de la tecnología de orientación a objetos.

Antes de pasar a discutir detalladamente aspectos del *Modelo de Proceso Flexible* definimos y analizamos, en el capítulo siguiente, los componentes de un modelo conceptual útil para la modelización de procesos.

3. Una propuesta de Modelo Conceptual para Modelado de Procesos

El objetivo de este capítulo es definir un modelo conceptual útil para comprender y soportar a la modelización de procesos. Para una mejor comprensión de esta parte, la estructuraremos del siguiente modo:

- primero realizamos una definición, bajo un enfoque descriptivo, de los diferentes conceptos que intervienen en procesos de software y modelado de los mismos, remitiendo al lector a distintas fuentes de la literatura investigada. El abordaje no pretende ser amplio (en cuanto a la cantidad de conceptos tratados[2]) sino el necesario para facilitar la comprensión del marco conceptual del punto 3.3.

- luego, presentamos un diagrama estático de las principales responsabilidades que intervienen en un proyecto de hipermedia (contextualizado para el modelo de proceso flexible).

- por último, presentamos un modelo conceptual para el modelado de procesos de software. Contar con una base que contenga un conjunto canónico de conceptos de modelado de procesos potencia, por una parte, la combinación de notaciones para describir a los procesos, y, por otra parte, posibilita la creación de capacidades en ambientes que soporten multi-paradigmas para ejecutar procesos.

3.1 Conceptos sobre Procesos de Software y Modelado de Procesos

Uno de los propósitos de esta tesis, según indicamos en el punto 1.1, es consolidar un modelo conceptual de clases y relaciones esenciales que sirva de base para investigaciones presentes y futuras. Dijimos que el conocimiento público de caracter científico, se ve favorecido si se

[2] No obstante, el lector cuenta con un glosario, el cual abarca mayor cantidad de conceptos y referencias que el que abordaremos en la siguiente sección

identifica y comunica, de un modo no ambiguo y representativo, a un conjunto de conceptos primitivos.

En ingeniería de procesos de software se manejan conceptos como el de proceso, tarea, actividad, agente, artefacto, recurso, rol, constructores de proceso, descripción de proceso, modelo de proceso, perspectivas de proceso, arquitectura de proceso, ambientes de ingeniería de software orientados a procesos, entre otros conceptos. Esfuerzos importantes se han venido realizando con el fin de establecer una base conceptual sólida [Curtis et al 92, Dowson et al 91, Feiler et al 93, Lonchamps 93, Madhavji 91]. Trabajos seminales en el área son los de Humphrey, Feiler y Dowson et al. Pensamos que nuestro modelo conceptual (fig. 3. 7), por medio de un diagrama de clases y relaciones, contribuye a representar y sintetizar de un modo no ambiguo el dominio del problema.

A seguir introduciremos los conceptos[3] usando un enfoque lexicográficamente descriptivo.

Una **fase** es una agrupación de *procesos de software* fuertemente relacionados o cohesivos realizados en cierto orden. En el esquema del *modelo de proceso* introducido en la sección 2.4 se puede apreciar tres fases: la de exploración, la de desarrollo y la operativa.

Humphrey ha dicho que un **proceso de software** (o proceso, para abreviar) es un conjunto parcialmente ordenado de *subprocesos* que tienen el fin de alcanzar alguna meta establecida. De un modo mas amplio podemos definir a un proceso de software como a un conjunto parcialmente ordenado de subprocesos a los que se le asocian una colección de *recursos*, *agentes*, *condiciones*, *artefactos* y *constructores de proceso*, con el fin de producir los *distribuibles* conforme a las metas establecidas.

Por **distribuible** entendemos a un artefacto (producto) solicitado por algún proceso o agente interno o externo.

Ejemplos de nombres de procesos a cierto nivel de granularidad son: administración de proyecto, determinación y especificación de requerimientos, análisis conceptual, diseño navegacional, diseño de interfaces abstractas, determinación y resolución de riesgos, análisis de factibilidad, prototipación, validación, etc.

[3] Utilizaremos la/s palabra/s resaltada/s en **negrita** para describir un nuevo concepto, con *itálica* los conceptos a describir en párrafos subsiguientes y, en esta sección, usaremos paréntesis para referirnos a (sinónimos).

Un proceso se puede descomponer en **subprocesos** (paso o elemento de proceso), por lo tanto le cabe una definición recursiva. El nodo raíz del árbol es el mas general y abstracto, el nodo hoja es el más específico y lo denominamos *actividad*. Por lo que una **actividad** es un subproceso que no requiere mas descomposición. Se dice que una actividad es un elemento de proceso descripto a un bajo nivel de granularidad (granularidad fina)

Es oportuno destacar que en la literatura se ha establecido una distinción, a veces sutil, entre el concepto de actividad y el de **tarea**. En el área de administración de proyectos una tarea es una unidad de trabajo a realizar por un agente. Una tarea es un subproceso a la que se le asocian componentes de gestión, es decir, se le pueden asignar agentes, recursos, se la puede planificar, programar, ejecutar y controlar.

En el área de ingeniería de procesos de software también se ha utilizado a la tarea como a la entidad atómica de abstracción, sobre todo en enfoques prescriptivos y orientados al análisis, como apunta Lonchamp. Sin embargo existen enfoques en que a las tareas se las dividen en actividades, siendo las actividades entes que no necesitan ser gerenciados (sobre todo en entornos o herramientas de automatización). Por lo tanto en el caso en que se establezca la diferencia entre tarea y actividad, ésta corresponderá a que un proceso pueda ser o no gerenciado por un agente. En la fig. 3.1 observamos la relación entre entes que representan a las definiciones previas de proceso, actividad y tarea.

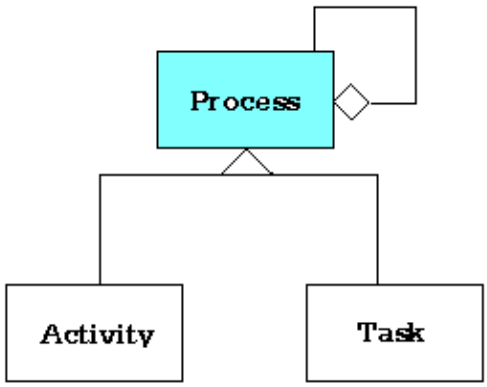


Fig. 3.1 *Diagrama que relaciona a las clases Proceso, Tarea y Actividad.*

En cuanto al concepto de **agente** lo definimos como al ente ejecutor de un proceso. El agente puede ser tanto un ente humano como una herramienta o dispositivo computarizado. Podemos realizar una taxonomía de tareas en función de los agentes teniendo en cuenta que una tarea se puede llevar a cabo por uno o mas agentes. Clasificamos a las tareas en tres tipos:

- manuales
- automáticas
- interactivas

Las tareas manuales solo involucran a agentes humanos; las tareas automáticas solo dependen de agentes computarizados, en tanto que las últimas necesitan de ambos tipos de agentes con el fin de alcanzar la submeta establecida. Para llevar a cabo un proceso el agente humano o computarizado debe contar con un conjunto de habilidades (experticia). Para cumplimentar un *rol* se deben requerir habilidades específicas por parte de los agentes. Por ejemplo, un agente humano debe tener experticia sobre criterios estéticos y cognitivos de interfaces de usuario y aspectos de diseño del look and feel para cumplimentar el rol de "Diseñador de Interfaces".

Un **artefacto** es el producto de realizar una tarea. Un artefacto es el producto creado, evolucionado, mantenido o destruido durante el proceso de desarrollo ya como un resultado requerido por un agente o para facilitar la prosecución de otro proceso. Con lo dicho podemos razonar que un artefacto puede servir de entrada a un proceso y, mediante la transformación correspondiente, ser la salida del mismo. Además un artefacto puede ser un objeto compuesto, es decir, se da una relación de agregación entre componentes.

En nuestra visión los procesos de software fundamentalmente existen para crear, modificar y reusar artefactos.

Entre los objetos considerados artefactos se encuentran los documentos -por ejemplo: documento del plan del proyecto, especificaciones de análisis y diseño, prototipos y versiones finales de productos -código fuente y/o ejecutable-. Los artefactos pueden ser administrados bajo una estrategia de configuración de cambios.

En cuanto a un **recurso** es un ente del mundo real necesario para que las tareas de un *proyecto de software* se puedan efectuar. Una tarea usa recursos. Ejemplos de recursos son: recursos humanos, monetarios, materiales, tecnológicos. En la figura 3.2 mostramos una jerarquía general de recursos usando una notación de [Rumbaugh et al 91].

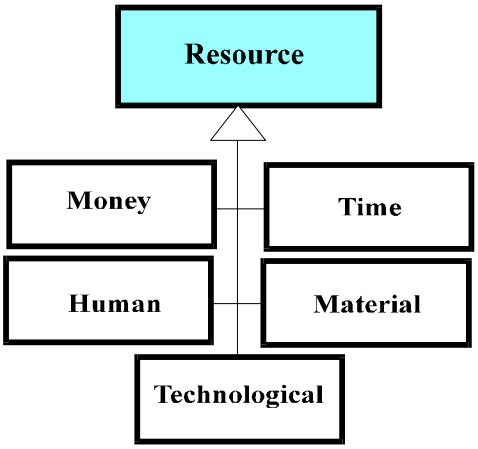


Fig. 3.2 *Diagrama general de jerarquía de recursos.*

Un **rol** es un conjunto de permisos y obligaciones que se debe asociar a un agente durante la realización de un tipo de tarea. El agente debe tener un conjunto de permisos para realizar las actividades de la tarea conforme a la submeta establecida y obligado a satisfacer un conjunto de *condiciones*.

Se puede establecer una jerarquía de roles: roles más generales y roles más específicos. En la figura 3.3 observamos que el rol de diseñador de interface es más específico que el rol de desarrollador de software.

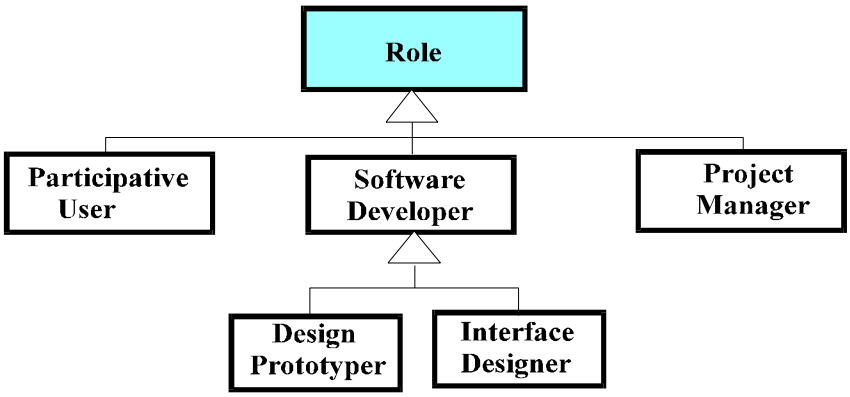


Fig. 3.3 *Ejemplo reducido de jerarquía de roles de un proyecto de hipermedia.*

En la página 493 de [Goldberg et al 95], los autores presentan un diagrama -de Venn- en donde se muestra los roles ocupados por agentes, agrupados estos en equipos de trabajo. Cuando un proceso se está llevando a cabo puede existir una asignación dinámica de roles a agentes y

viceversa. Los agentes pueden ocupar (jugar) diferentes roles a diferentes momentos, un rol dado puede ser ocupado por distintos agentes a diferentes momentos [Dowson et al 91].

Una **condición** de un proceso es la declaración del estado de situación que debe acontecer para el inicio, ejecución y finalización de un proceso. Una actividad puede comenzar cuando se cumple un conjunto de precondiciones y puede finalizar cuando se alcanzan las postcondiciones establecidas.

Un **constructor de proceso** es un enfoque específico (método) que se puede usar para realizar una tarea. Ampliando el concepto podemos decir que para llevar a cabo una tarea especificada en la *descripción de proceso* se pueden usar diferentes métodos que responden a distintos enfoques (ver fig. 3.4).

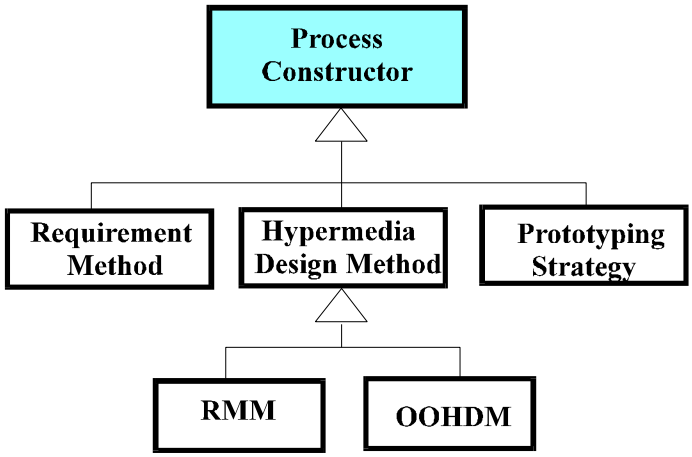


Fig. 3.4 *Ejemplo reducido de jerarquía de constructores de proceso de un proyecto de hipermedia.*

Por ejemplo, para una tarea de "modelado del dominio de la aplicación", podemos usar el método especificado en RMM, denominado "diseño de E-R", o podemos usar para esa tarea el método especificado en la metodología OOHDM, denominado "modelado conceptual". Son enfoques diferentes para una descripción de proceso dada.

En el trabajo seminal sobre el área de ingeniería de procesos, Osterweil dijo [Osterweil 87] que los procesos de software son también software. Si bien el enfoque era formal, centrado en descripciones procedurales y declarativas, hay diferentes paradigmas para describir a los procesos. Primero veamos el concepto.

Una **descripción de proceso** es una manera de representar y especificar la secuencia parcial de actividades de un proceso. Una descripción completa de proceso debe considerar las actividades y las operaciones asociadas, las precondiciones y postcondiciones para cada actividad, y otros entes (objetos) intervinientes en el proceso de software como artefactos, agentes y roles. Pueden coexistir descripciones alternativas para un proceso y se puede considerar a una descripción de proceso como a un artefacto especial.

Una descripción de proceso está para que pueda ser interpretada y ejecutada por un agente humano o un agente computarizado. Un área de investigación atractiva es la de soporte a procesos de software. Se han realizado algunas contribuciones y avances, en el área de *ambientes de ingeniería de software centrado en procesos*. En estos sistemas muchas de las tareas son interpretadas y ejecutadas automáticamente por un *motor de procesos*. Puede coexistir para una cierta actividad descripciones alternativas bajo un mismo enfoque, como indicamos anteriormente, o descripciones multi-paradigmáticas.

Los aspectos de mayor o menor formalidad en la descripción de procesos es una cuestión de relevancia. Una representación precisa, formal, es ejecutable por una máquina, en tanto que una representación informal, esto es en lenguaje natural, o una representación semiformal estructurada en un script, puede ser ejecutada por un agente humano y no por un agente computarizado - aunque en algunos casos una representación semiformal puede ser interpretada.

En la figura 3.5 el lector puede observar una taxonomía de descripciones de procesos en función de la formalidad de su representación. Otra taxonomía se podría realizar o complementar teniendo en cuenta el nivel de granularidad de las actividades descriptas. Los autores en [Curtis et al 92], han expresado que la experiencia sugiere que los procesos automatizados requieren una representación de granularidad más fina, en tanto que una representación semiformal como los script, de granularidad más alta, son adecuadas para ser interpretadas y ejecutadas por agentes humanos.

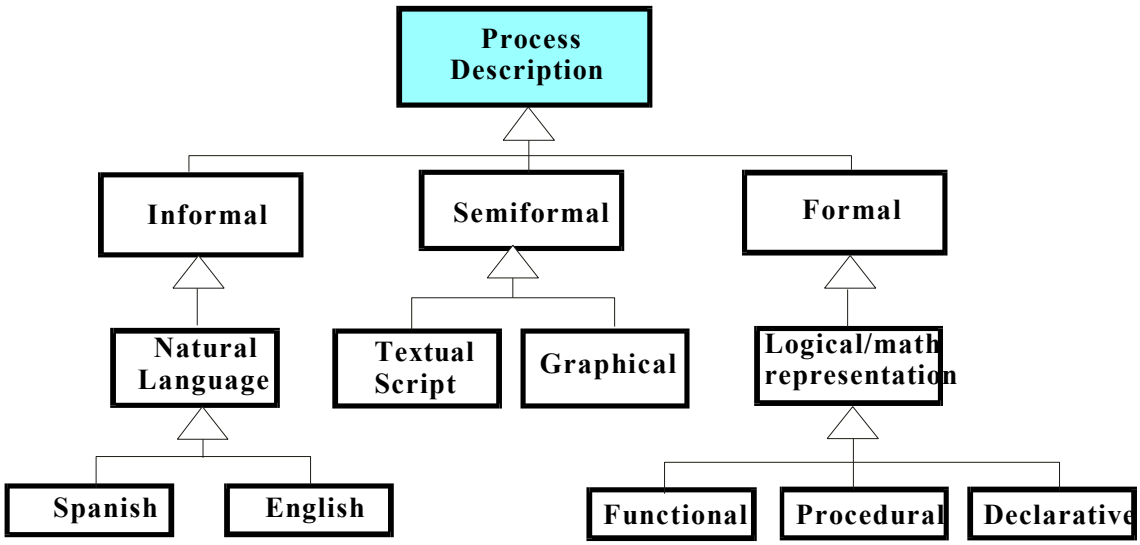


Fig. 3.5 *Diagrama reducido de tipos de descripciones de proceso en función de la formalidad de su representación.*

Relacionado a la representación de las descripciones de procesos y a las distintas *perspectivas* de un *modelo de proceso* está el concepto de *lenguajes de modelado de procesos*, que describiremos en párrafos subsiguientes.

Los procesos de software se pueden abstraer en un **modelo de proceso** (modelo de ciclo de vida). Definimos a un **modelo** como a una representación abstracta de entes o fenómenos de la realidad en la que se consideran los aspectos relevantes de los mismos y se desechan los menos relevantes sin que por ello deje de representar significativamente a esa realidad.

Definimos a un modelo de proceso de software como a una estrategia apropiada para abstraer, organizar, ejecutar y/o controlar a las distintas fases, tareas, recursos y artefactos de un proyecto con el objeto de alcanzar las metas establecidas. Un modelo de proceso es una descripción más o menos formal del proceso de desarrollo de software. Por lo que un modelo de proceso expresa: 1) un cierto nivel de abstracción; 2) una *perspectiva* particular del proceso de desarrollo.

Modelos de ciclo de vida como los discutidos en la sección 2.2 -modelo de cascada, espiral y recursivo/paralelo-, son modelos muy abstractos y generalmente semiformales. No se especifica con suficiente detalle y rigor formal, las entradas y salidas de las actividades, condiciones de comienzo y finalización, flujos de artefactos, sincronización de actividades, entre otros asuntos.

Un modelo de proceso integra diferentes tipos de información de los procesos de software. Una **perspectiva** (vista o submodelo) es un enfoque particular para especificar y comunicar información del modelo. En la sección 2.3 dijimos que un modelo de proceso debe ayudar a responder preguntas tales como: qué hacer, cómo y cuándo hacerlo, dónde y quiénes lo harán, qué dependencias existirán entre tareas y cómo se sincronizarán, etc.

Curtis et al agrupan la información en cuatro perspectivas con el fin de abstraer la complejidad subyacente en los procesos de software, y son, a saber: *perspectiva funcional*, *perspectiva informacional*, *perspectiva de comportamiento* y *perspectiva organizacional.* Pensamos que una *perspectiva metodológica* es fundamental para representar a los diferentes métodos (constructores de proceso) y, además, puede ser integrado en *ambientes de ingeniería de software centrado en procesos.*

La **perspectiva funcional** representa qué tareas se deben realizar en cada fase, qué estructura jerárquica de actividades existe para cada tarea, y qué dependencias funcionales -flujos de información y control, documentos, artefactos-, son relevantes para las entradas y las salidas.

La **perspectiva de información** se centra en los artefactos producidos o requeridos por los procesos de software; en la estructura de los artefactos y sus interrelaciones; en la estrategias de administración de cambios de artefactos y seguimiento (traceability) de los mismos.

La **perspectiva de constructores de proceso** representa los métodos a aplicar para efectuar las distintas actividades especificada en la descripción de procesos. Para un método en particular puede haber una o más herramientas que den soporte a la tarea.

La **perspectiva de comportamiento** representa aspectos dinámicos del modelo de proceso: el secuenciamiento y la sincronización de las tareas; información de cómo se realizan las actividades, esto es, iteraciones, ciclos de retroalimentación, paralelismo, condiciones de inicio y terminación, entre otros asuntos. Asimismo puede especificar el ciclo de vida de un ente como un artefacto, proceso, agente con formalismos como: diagrama de transición y estados, statecharts, ADV-charts, etc.

En la **perspectiva organizacional** se consideran aspectos de qué agentes participan/planifican/ejecutan y controlan a qué tareas; qué roles se asignan a los agentes participantes; qué estrategias de comunicación y de dinámica de grupos se aplican, etc.

Por lo tanto, un modelo de proceso integral y completo, necesita de una **arquitectura de proceso** (meta-modelo de proceso) para conectar a las diferentes perspectivas. Una arquitectura de proceso es una estructura conceptual útil para describir las perspectivas relevantes y sus relaciones.

Un **lenguaje de modelado de procesos** es un paradigma capaz de representar a un esquema de meta-modelo de proceso. Como todo lenguaje debe proveer una sintaxis precisa y una semántica no ambigua y amplia. Desafortunadamente, es difícil contar con un lenguaje que represente a las distintas perspectivas de un modelo de proceso y provea constructores de soporte a las distintas actividades de modo que pueda ser interpretado y ejecutado con eficacia en diferentes *proyectos de software.* En las práctica, la mayoría de las descripciones de proceso, cuando existen, se basan en mecanismos informales o semiformales difíciles de ser ejecutados en *ambientes centrados en procesos*.

No obstante algunos aspectos de un modelo de proceso pueden ser automatizados. Curtis et al presentan un conjunto de tipos de lenguajes que responden a distintos paradigmas, que pueden ser útiles para representar a las diferentes perspectivas de un modelo de proceso. Si bien este trabajo requeriría ser actualizado, creemos que es válido para obtener una visión inicial del estado del arte.

Un **ambiente de ingeniería de software centrado en procesos** (ambiente (o sistema) de soporte a procesos de software) es un entorno de trabajo que ofrece asistencia a los usuarios en la ejecución de un *proyecto de software*. Un componente esencial del ambiente es el **motor de procesos** el cual es el intérprete automático de las descripciones de proceso provistas de un modo estático o dinámico. Con dinámico queremos significar que durante la ejecución del proceso la descripción puede cambiar (evolucionar).

La arquitectura de un ambiente de software centrado en procesos debe 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.

La realización de un **proyecto de software** comprende a un conjunto de tareas, tanto técnicas como de gerenciamiento, a un conjunto de recursos, a un conjunto de estrategias, métodos y heurísticas, con el propósito de lograr los objetivos y las metas acordadas. Un proyecto de

software puede ser autocontenido o parte de otro proyecto mayor y está inserto en el marco de alguna estrategia organizacional.

3.2 Principales componentes de un Proyecto de Software de Hipermedia

En la figura 3.6 representamos con un diagrama de clases, relaciones y subsistemas, los componentes intervinientes en un proyecto de software de hipermedia. El fin del diagrama es agrupar un conjunto de responsabilidades y colaboraciones -usando una notación ampliada a la de [Rumbaugh91], y representar de un modo estático al proyecto de hipermedia en el contexto del modelo de proceso flexible.

Se puede apreciar los siguientes componentes:

1. el de *modelo de proceso* el cual representa o abstrae al proyecto. Un proyecto de software de hipermedia debe adoptar un modelo de ciclo de vida (en caso de no adoptar uno se guiará por una estrategia ad-hoc).
2. el módulo de *tarea* que representa a la unidad fundamental de gerenciamiento de un proyecto. Una tarea se la puede planificar, programar, ejecutar y controlar.
3. *el de meta y objetivos*, que representa a un conjunto de declaraciones de los resultados que se desean alcanzar en el contexto de la estrategia organizacional.
4. *el de recurso*: tecnológicos, humanos y otros. Los primeros están conformados por los componentes de software y hardware, los que automatizan parcialmente fases y tareas; los segundos por los agentes humanos, los que se le asocian diferentes roles en el transcurso del proceso. Desde otro punto de vista el proyecto está sujeto a restricciones de recursos monetarios, materiales, humanos, etc.
5. el de *constructor de proceso* para las distintas fases, tareas y actividades. Para un proyecto de hipermedia se requiere genéricamente: un modelo de Plan del Proyecto, un método para Análisis de Requerimientos, un método de Diseño, una estrategia de Constucción e Integración, criterios Cognitivos y Estéticos para la construcción de interfaces y espacios de navegación, una estrategia de Aseguramiento de la Calidad.
6. el *repositorio de artefactos de software* el cual sirve de entrada y de salida a las tres fases del modelo de proceso flexible (que discutiremos en el capítulo 4). Los artefactos del proyecto debieran estar sujetos a una estrategia de Configuración de Cambios, de todos aquellos ítems que se establezcan como configurables.

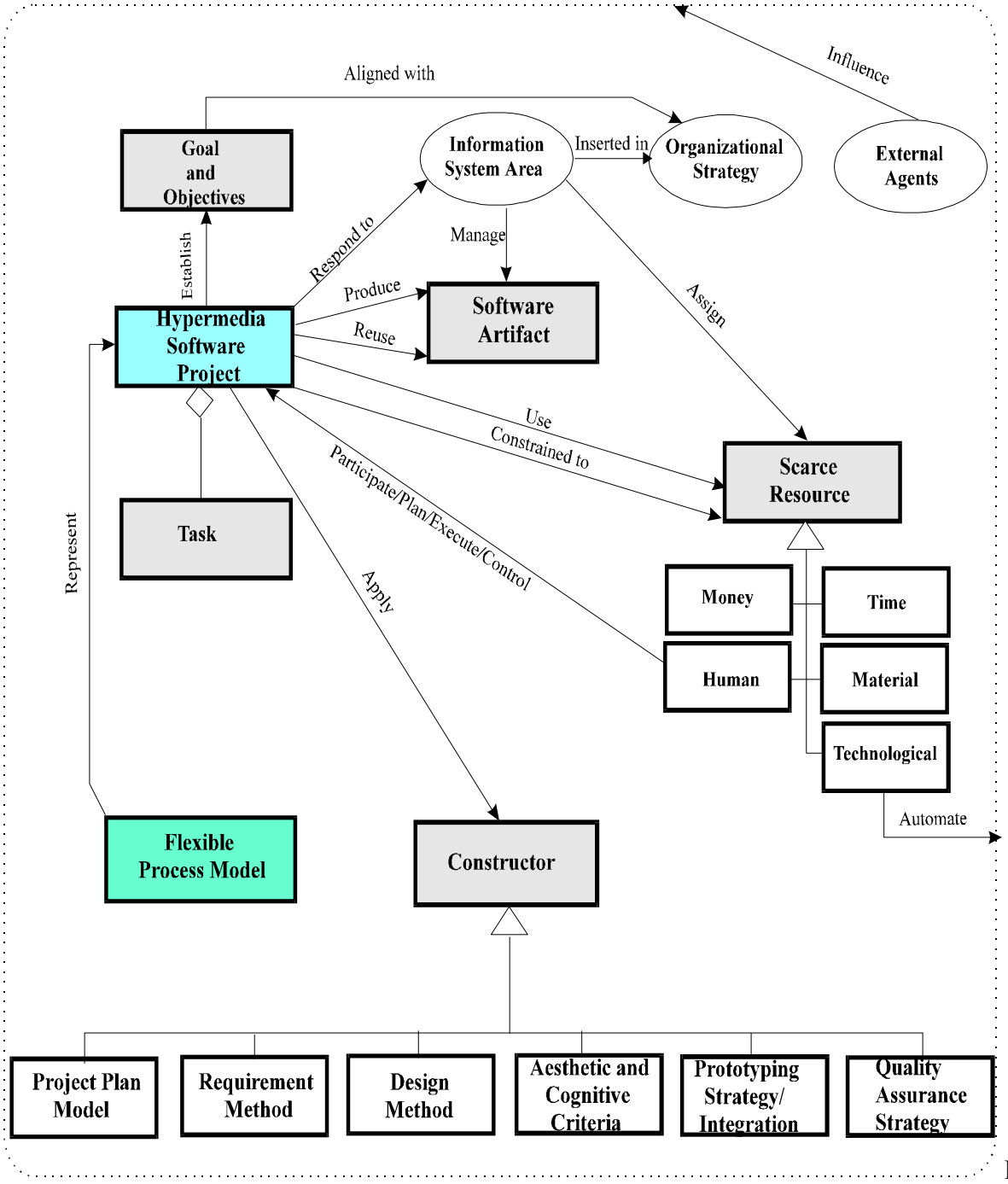


Fig. 3.6 *Visión estática de un proyecto de hipermedia como parte del Modelo de Proceso Flexible.*

3.3 Definición de un Modelo Conceptual

En esta sección presentamos un modelo conceptual para el modelado de procesos de software. Pensamos que al definir un marco base que contenga a un conjunto canónico de conceptos que intervienen en modelado de procesos, potencia a las investigaciones en el área, al permitir:

- la posibilidad de combinar diferentes clases y relaciones para dar soporte a las distintas perspectivas.

- la posibilidad de combinar formalismos y/o notaciones para describir a los procesos que, en definitiva, propenderá a la estandarización de procesos (sección 1.1).

- la posibilidad de crear capacidades en ambientes de ingeniería de software centrado en procesos para que soporten multi-paradigmas en la ejecución de actividades.

Del análisis de la literatura investigada sobre el área, y de las observaciones iniciales efectuadas en los proyectos de desarrollo de hipermedia en que hemos participado, se puede indicar que:

- no existe una única notación universal para especificar a todos los aspectos de un modelo de proceso.

- como consecuencia se requiere un lenguaje multi-paradigma para describir y especificar a los distintos entes y relaciones. El ambiente debiera soportar los diferentes tipos de descripción de procesos (formales, semiformales e informales), para ser ejecutados por agentes humanos o automáticos.

- Dada la complejidad de los elementos y fenómenos que intervienen en un proyecto de software, es importante centrar la atención en las perspectivas del modelo de proceso para ofrecer facilidades de ambiente.

- Un ambiente debe proveer capacidad de redefinición dinámica de descripciones para soportar aspectos evolutivos del proceso de desarrollo.

Algunas de las hipótesis e indicaciones previas serán motivo de investigaciones futuras.

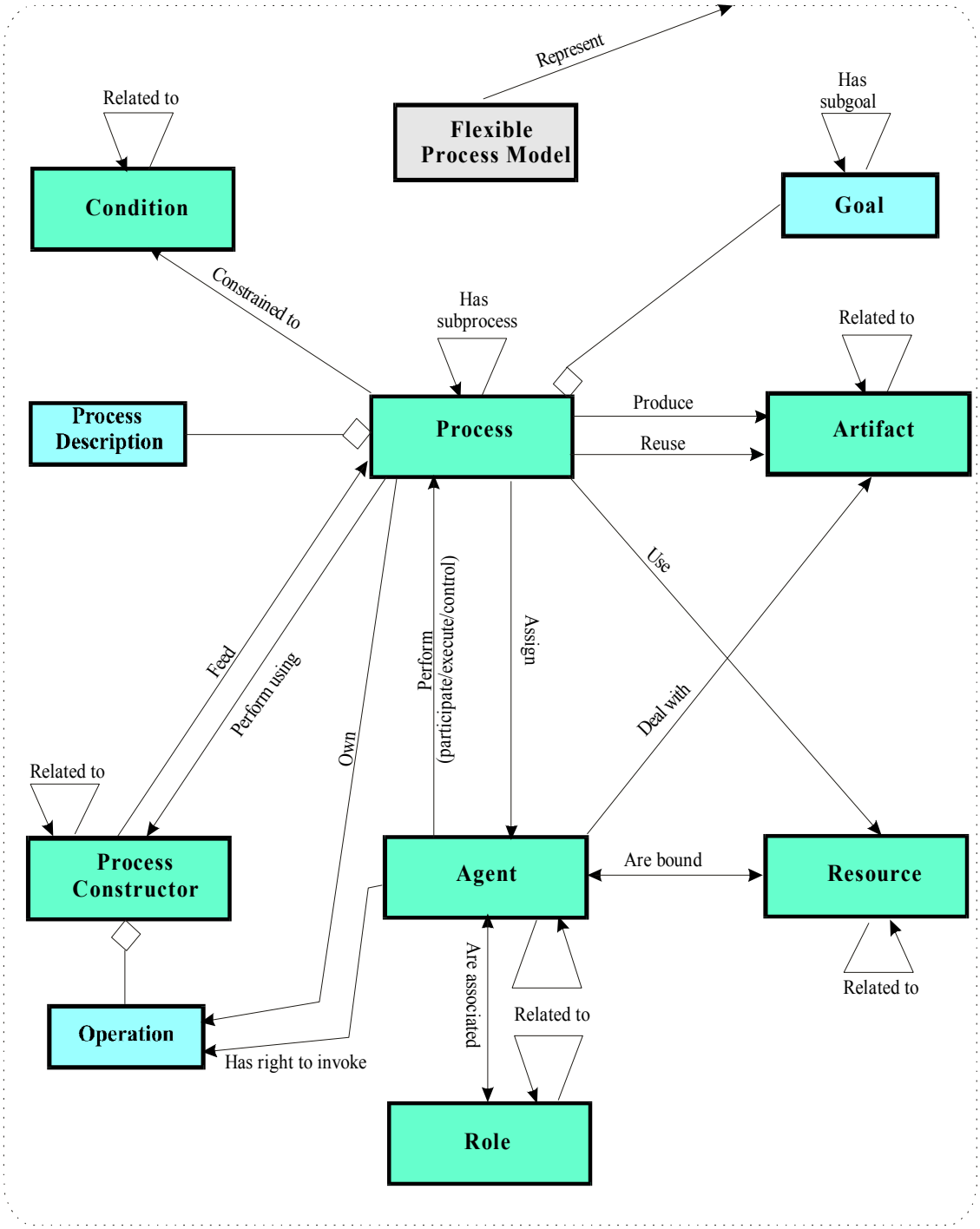


Fig. 3.7 *Diagrama de clases y relaciones fundamentales para un Modelo de Proceso Flexible.*

El objetivo de esta parte del trabajo consiste en presentar un modelo conceptual especificado mediante un diagrama de clases y relaciones fundamentales (figura 3.7), soportado por los

principios de Orientación a Objetos.

Las clases (o conceptos clave para nuestro dominio del problema que es el modelado de proceso) representan comportamiento y estado. En definitiva disponen de métodos para la ejecución de operaciones y encapsulan atributos del ente. Existen métodos denominados constructores y destructores entre otro tipos de métodos. Por ejemplo una clase artefacto puede tener como atributos su identidad, su fecha de creación, y terminación (para una versión dada), el identificador del documento asociado, etc. y puede responder a solicitudes de un agente por medio de operaciones de creación, destrucción, modificación del contenido o de sus atributos.

Por otra parte, las clases soportan diferentes mecanismos de relaciones [Booch 94, Rumbaugh et al 91]. Las relaciones pueden darse entre entes de una misma clase, o entre clases distintas. Podemos citar mecanismos de herencia, de agregación, de asociación. Por ejemplo en la fig. 3.3 mostramos un diagrama simplificado de la jerarquía de herencia para la clase "Rol"; un artefacto compuesto está relacionado por un mecanismo de agregación; las clases fundamentales se relacionan entre sí por medio de mecanismos de asociación (y uso, como caso particular de la asociación).

Por ejemplo en la fig. 3.7 una clase de tipo "Agente"[4] asociada o jugando un tipo de "Rol" realiza un tipo de "Tarea". (Por ejemplo: "un agente, Gustavo" controla "la tarea, diseño de contextos de navegación" cuando ocupa el "rol, coordinador del proyecto"; "un agente, Luis" ejecuta "la tarea, diseño de contextos de navegación" cuando ocupa el "rol, desarrollador del proyecto"; "un agente, Marina" participa "en la tarea, diseño de contextos de navegación" cuando ocupa el "rol, usuario participante". El lector puede observar que los agentes tienen distintos derechos, y que se puede establecer un mecanismo de delegación).

En la figura 3.7 consideramos explícitamente las relaciones fundamentales entre las clases. Sin embargo con siete clases bases podríamos tener un número de relaciones potenciales de siete factorial (7!). Por ello una división de preocupaciones por perspectivas puede disminuir la complejidad en el modelado de procesos.

3.3.1 Descripción de las Clases y sus Relaciones Fundamentales

Si bien en la sección 3.1 introducimos un conjunto de conceptos, a seguir profundizaremos

[4] a seguir subrayaremos a las relaciones y pondremos entre comillas a las clases y objetos (en algunos casos acompañada de otra información).

sobre los entes y las relaciones mostradas en la fig. 3.7

Una “tarea” representa una unidad de trabajo que se le asigna a un “agente” para su realización. Una “tarea” contiene “descripciones de proceso”, las que pueden comprender una colección de alternativas -instanciables-, para representar a la misma unidad de trabajo. Asimismo pueden estar especificadas en diferentes formalismos y/o notaciones (fig. 3.5), las que serán comprensibles para los distintos “agentes” involucrados.

Si observamos (leemos) algunas clases y relaciones del diagrama podemos decir que un “agente” realiza una “tarea” contenida en la “descripción de proceso”, usando algún “constructor de proceso” en particular. Para la efectivización de la tarea el “agente” tiene el derecho a invocar las “operaciones” que son parte del “constructor”. Esta clase al ser usada alimenta al “proceso” cuya finalidad es producir un “artefacto” que esté dentro de las “metas” establecidas. El “proceso” se lleva a cabo bajo ciertas “condiciones”.

Además de la “operaciones” que posee una “tarea” asociada al constructor de proceso en particular, la tarea también posee métodos para crear una actividad (constructor), y para la terminación de la misma (destructor). Por otra parte, hay métodos para la delegación de actividades.

Dado el caracter jerárquico de las tareas, en la que una actividad puede depender de otras actividades, delegar una actividad al agente correcto implica que éste tendrá el derecho de invocación de ciertas operaciones. La idea de delegación surge de la relación jerárquica de actividades y en los permisos de los agentes a controlar, ejecutar y/o participar en el desarrollo de las tareas. De este modo podemos distinguir entre el concepto de que una tarea posee operaciones y el concepto de que el agente tiene el derecho a invocarlas. (Observar el ejemplo de Gustavo, Luis y Marina, dado mas arriba, para ejercitar estas ideas).

Un artefacto es un objeto persistente que representa el producto de realizar una tarea. Esto es, un “proceso” produce “artefactos” por medio de la colaboración de uno o más “agentes”, y a su vez un “proceso” reusa “artefactos”.

Un artefacto puede ser un objeto simple o compuesto y puede estar sujeto a un sistema de configuración de cambios. El mismo consiste en una estrategia -asociado a algún método- para identificar, mantener, y administrar los cambios de los artefactos bajo cierta configuración. Se debe considerar aspectos de procedimientos de aceptación y congelamiento de los cambios,

ítems configurables, versionamiento, entre otros asuntos.

Desde el punto de vista de un sistema de configuración y de un modelo de seguimiento, un artefacto es un objeto con propiedades y comportamiento visibles. Un artefacto es un objeto persistente que se caracteriza por su identidad, tipo de artefacto (por ej. un documento es un tipo de artefacto- ver la definición en el glosario), fecha de creación, de ultima actualización, versión, estado (por ej. en estado de procesamiento o en estado de aprobación). Además los agentes, humanos y computarizados, disponen de un conjunto de servicios para realizar sus acciones. Una taxonomía de métodos podría ser la siguiente:

- métodos de contenido, para operar sobre el contenido de un objeto (en general se dispone de herramientas específicas asociadas a constructores de proceso)
- métodos estructurales, para crear nuevos objetos, para agregar a la jerarquía existente, para eliminar artefactos, para consultar la estructuras de relaciones
- métodos para operar sobre el estado
- métodos para dar soporte a un modelo de seguimiento

Considerando una perspectiva de comportamiento, y centrándonos solamente en un modelo de ciclo de vida de un artefacto, podemos utilizar formalismos de modelado tradicionales como, máquinas de estado finita, cartas de estado (statecharts) y sus derivados o redes de Petri. Un posible esquema de transiciones y estados para el ciclo de vida de un ente [Humphrey et al 89], en nuestro caso un artefacto, es el que mostramos en la figura 3.8.

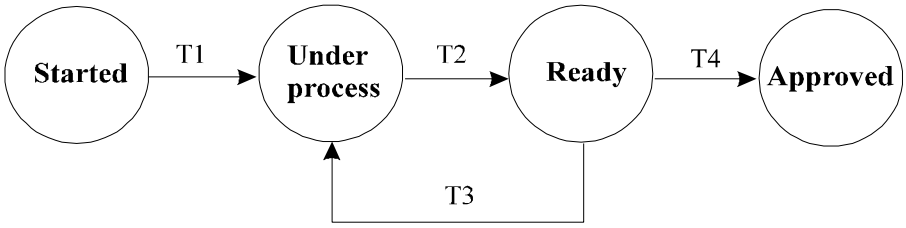


Fig. 3.8 *Un diagrama de transiciones y estados que representa el ciclo de vida de un artefacto.*

Una máquina de estados asociada a un artefacto puede facilitar la definición de la semántica de los estados y transiciones de un producto. Por ejemplo para un artefacto compuesto un objeto de la jerarquía puede pasar a estado “aprobado” si todos sus componentes están aprobados y pasaron a conformar la línea base del sistema de configuración y administración de cambios.

En cuanto a las transiciones son disparadas por eventos. Para que una transición provoque un cambio de estado, se deben cumplir las precondiciones establecidas; el efecto del cambio se puede representar en las postcondiciones. Una forma de especificar lo anterior es mediante la siguiente notación:

Transición $\mathbf{T_n}$: <*Evento, Estado previo, Estado siguiente, Precondición, Postcondición*>

Un ejemplo de condición para pasar del estado "listo" al estado "aprobado" de un objeto compuesto es el indicado más arriba; otra condición es que el agente que dispara la transición, tenga el permiso apropiado (en el cumplimiento de su rol) para provocar los cambios de estado de "listo" a "aprobado".

Finalmente, en cuanto a la clase "agente" abstrae al ente que controla y/o ejecuta y/o participa en la realización de las "actividades" y, además, se relaciona con la clase "artefacto".

La clase "agente" tiene una relación muchos a muchos con la clase "recurso". En la figura 3.2 mostramos un diagrama general de tipos de recursos. Un recurso se puede asociar a varios agentes y, un agente puede estar asociado o usar varios recursos. Por ejemplo una persona, que es un recurso, puede asociarse a varios agentes, y un agente puede usar varios recursos como, una persona, hardware y sistema operativo de una computadora, espacio físico, etc. Un agente puede usar una composición de recursos del mismo tipo (equipo de personas).

3.3.2 Comentarios finales

Esta división de responsabilidades entre la clase "agente" y la clase "recurso" es relevante para los objetivos de la modelización. Se puede pensar en la ejecución de una actividad sin que estén disponibles los recursos. Estos representan a objetos del mundo real, en tanto que los agentes representan a objetos del mundo lógico -que en definitiva implica mayor nivel de abstracción.

La división de preocupaciones entre las clases "agente", "recurso" y "rol" es también de importancia. Principalmente a la separación entre un agente abstracto asociado a un rol para realizar una actividad, de un agente específico, con habilidades particulares. Por ejemplo, la sentencia "El agente X -uno no instanciado-, está asociado al rol de desarrollador del proyecto, cuando realiza el subproceso de diseño navegacional", es de un nivel más abstracto e independiente de la alocación de recursos y ocupación de roles que la siguiente sentencia: "El

agente Gustavo, ocupa el rol de desarrollador del proyecto, dado que posee experticia reconocida en la intervención de los proyectos Menphis y Pampa I, para la realización de la tarea de diseño navegacional, en el proyecto Pampa II". Aquí el ocupante del rol, que es un recurso específico provisto por la organización, posee habilidades específicas, para que integre el nuevo proyecto de hipermedia.

La definición de un modelo conceptual y la distinción entre distintos niveles de abstracción y de preocupaciones son, como indicamos anteriormente, de importancia para la modelización de procesos por varios motivos:
Primero, porque representa a las abstracciones claves del dominio del problema, agrupando comportamientos comunes y haciendo explícitas un conjunto de relaciones. Esto favorece la distribución de responsabilidades e identificación de colaboraciones.

Segundo, favorece el esquema de perspectivas, es decir, la división de preocupaciones en submodelos o vistas del modelo de proceso. Como describimos en la sección 3.1, para disminuir la complejidad, es conveniente separar diferentes tipos de información de los procesos para especificar, comunicar y controlar porciones del modelo. Definimos las perspectivas funcional, informacional, de comportamiento, organizacional y metodológica. Por ejemplo, una perspectiva funcional se focaliza principalmente en la clase proceso y sus relaciones; una vista informacional se focaliza en la clase artefacto, sus componentes y relaciones; una vista organizacional se concentra en las clases recurso, agente y rol y la asignación a las distintas tareas.

Por último, y desde otro punto de vista, es importante la separación de preocupaciones de estos dos enfoques para la modelización: aquel enfoque que no depende de los recursos específicos ni de las diferentes instancias de constructores a usar para una descripción de procesos, de aquel otro enfoque que se concentra esencialmente en los recursos específicos, habilidades particulares de los agentes y ocupación de roles. La figura 3.9 muestra un diagrama representativo del primer enfoque.

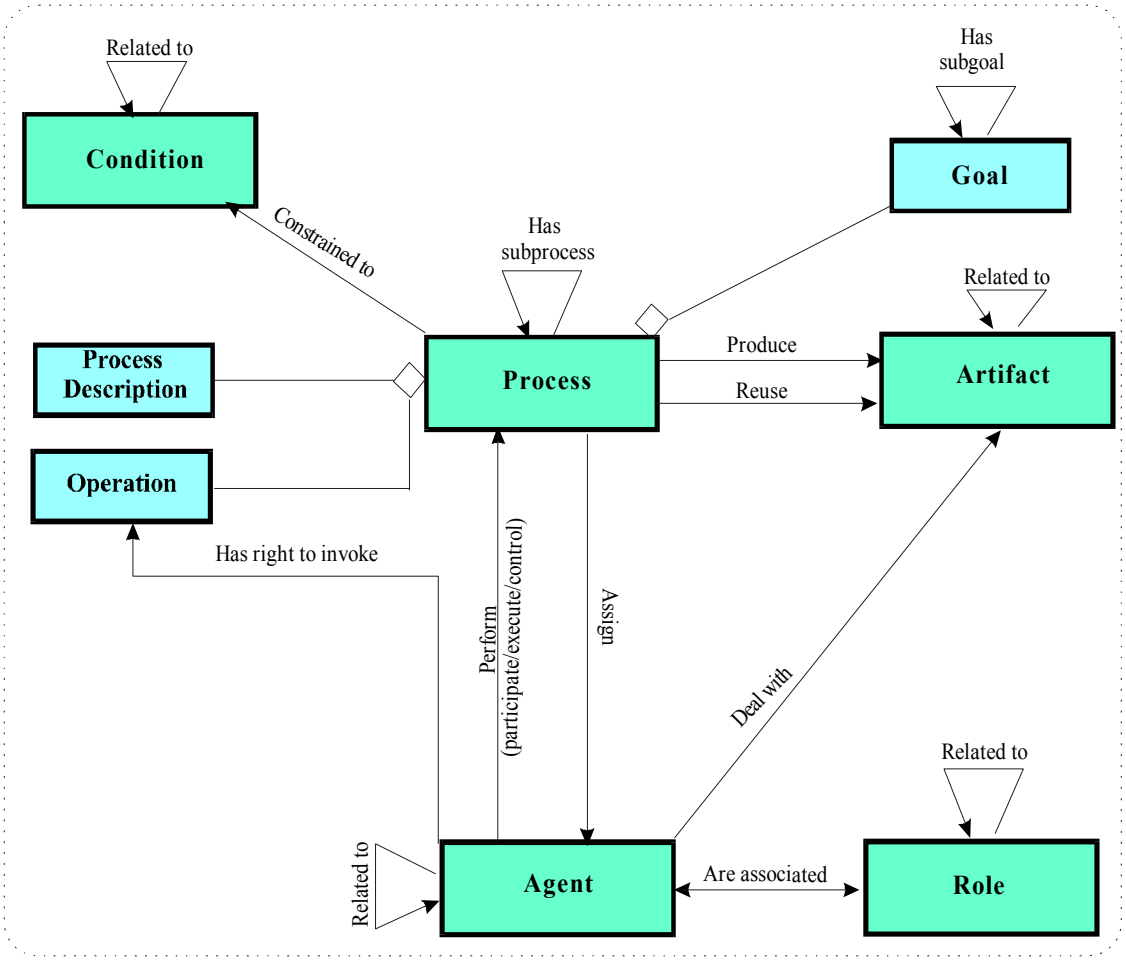


Fig. 3.9 *Diagrama de clases y relaciones fundamentales para un enfoque independiente de los recursos y constructores de proceso.*

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.

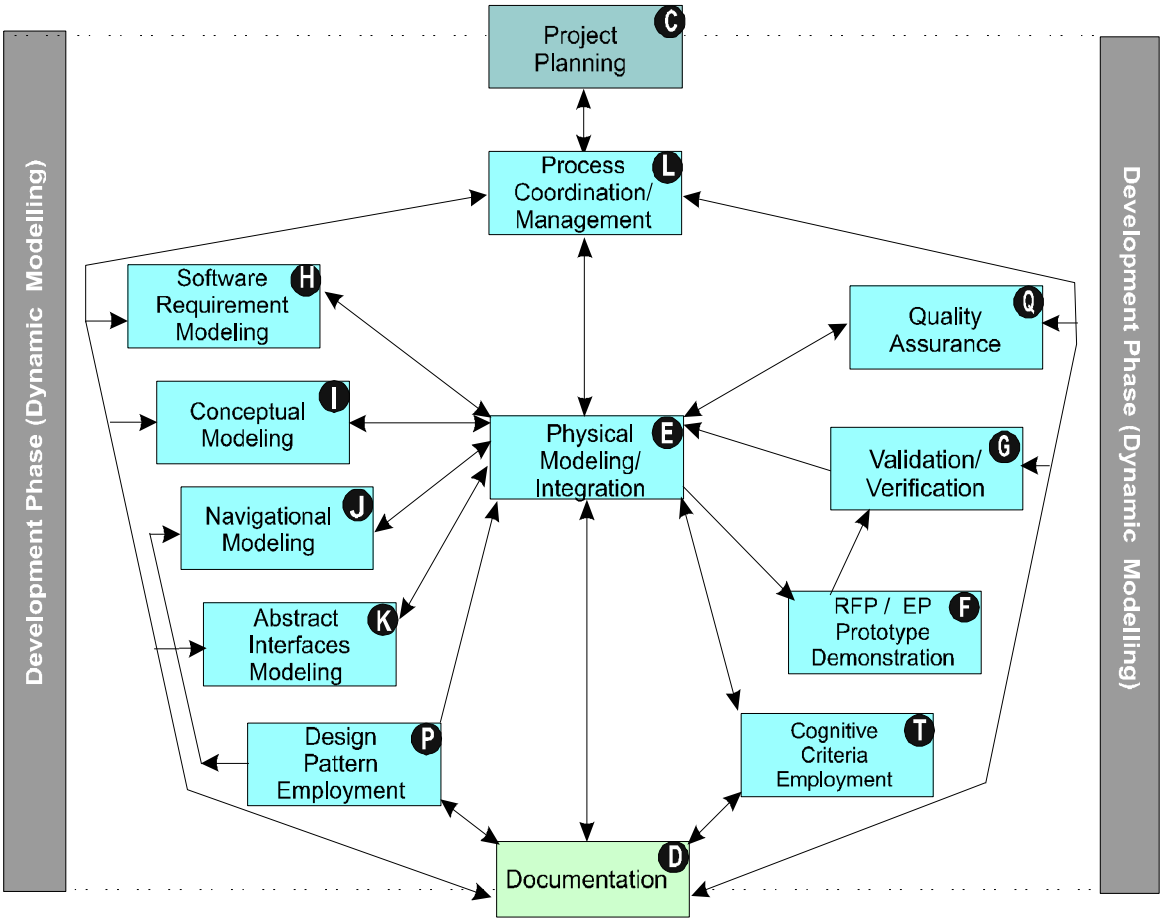


Fig. 4.1 *Principales tareas de la fase de desarrollo*

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.).

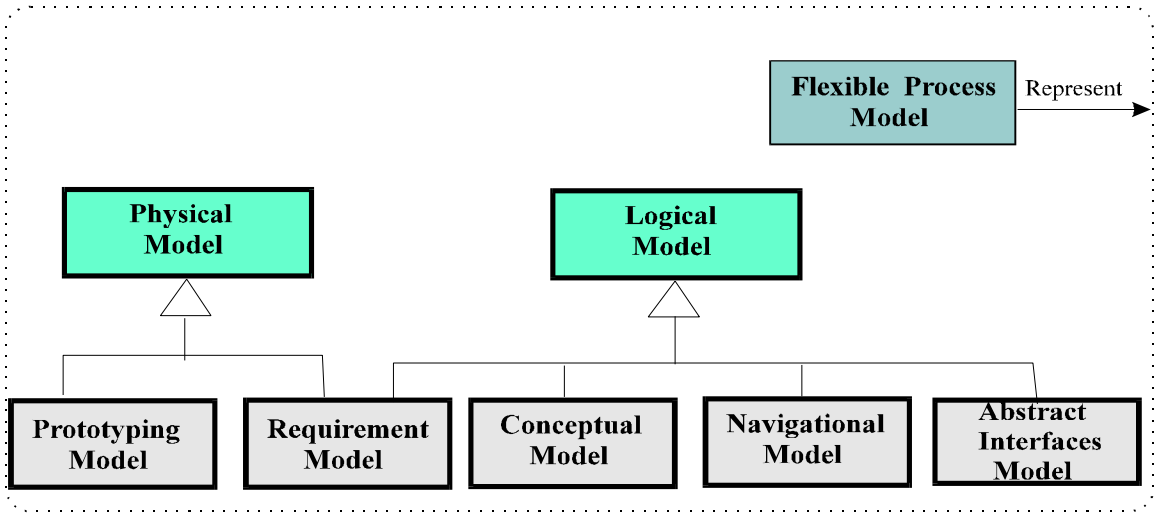


Fig. 4.2 *Algunos modelos lógicos y físicos a utilizar en el proceso de desarrollo*

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.

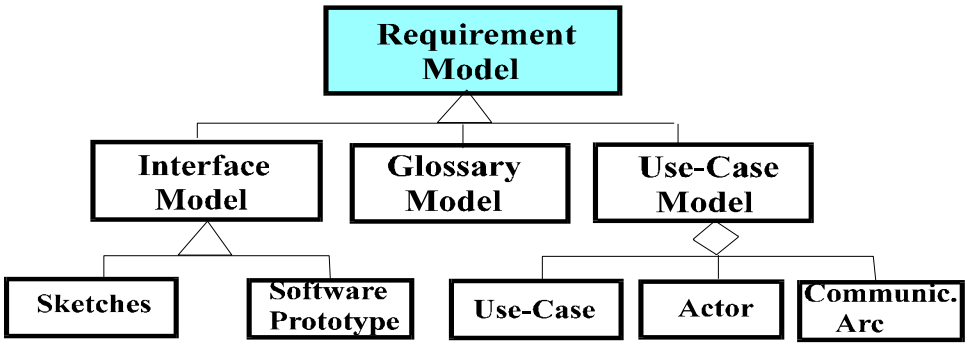


Fig. 4.3 *Tres modelos que conforman al modelo de requerimentos*

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).

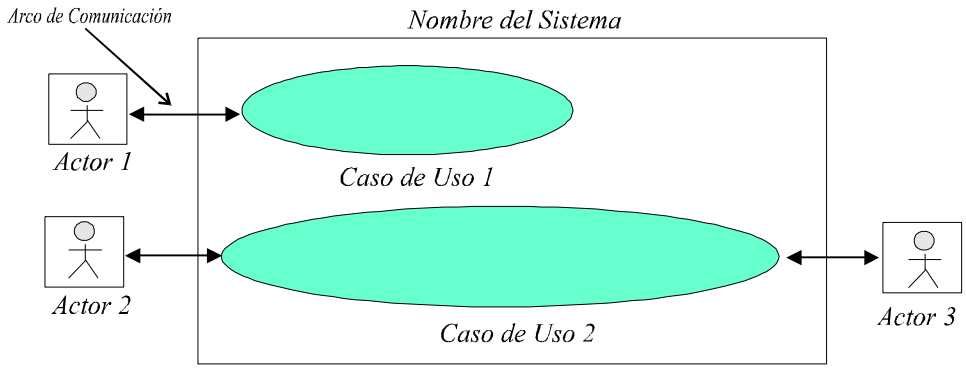


Fig. 4.4 *Esquematización de los elementos del modelo de casos de uso*

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.

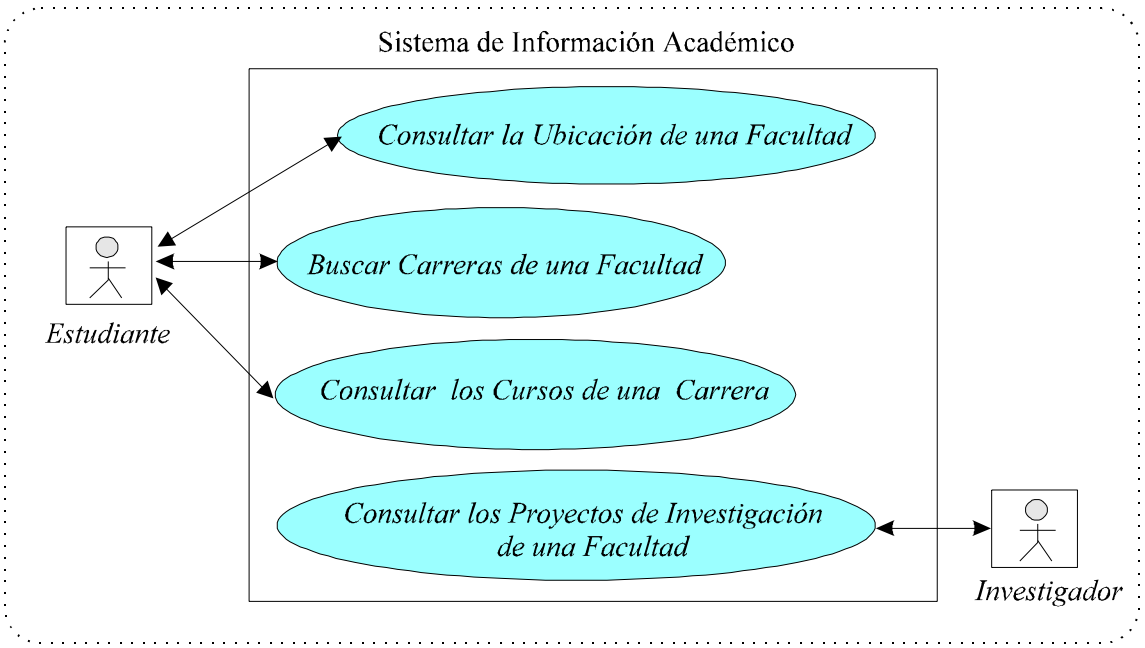


Fig. 4.5 *Casos de uso reducido de un SIA*

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.

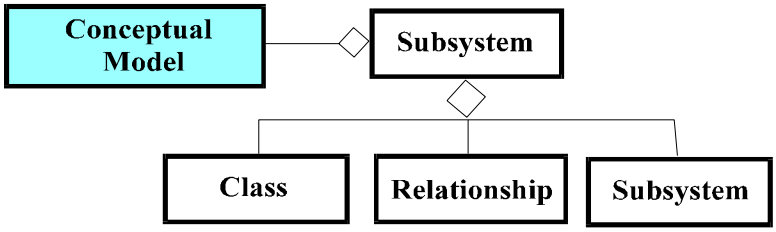


Fig. 4.7 *Diagrama de composición de las primitivas del modelo conceptual*

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].

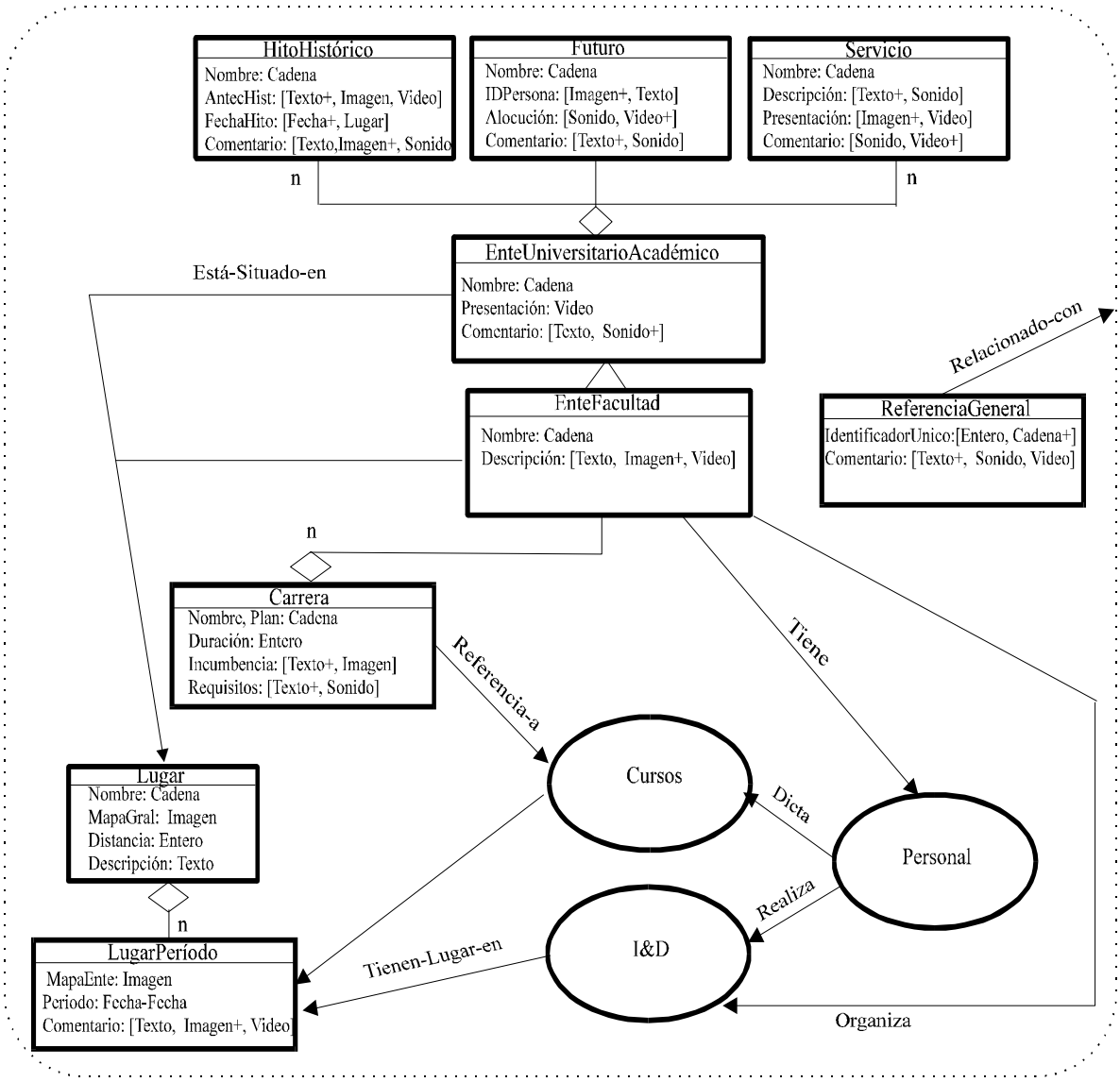


Fig. 4.8 *Diagrama de clases, relaciones y subsistemas del modelo conceptual del SIA*

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.

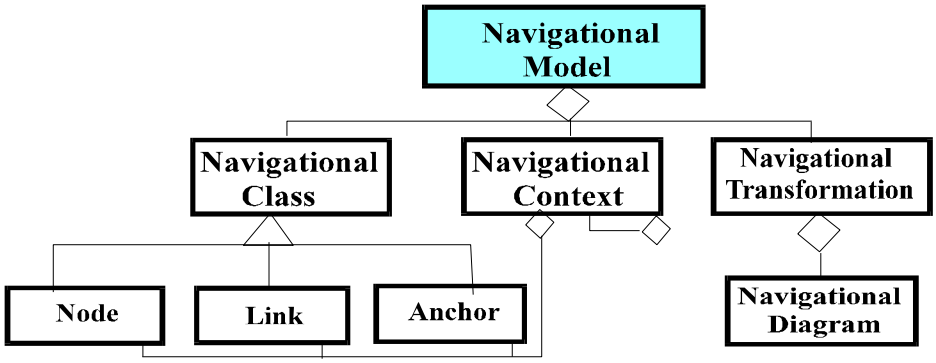


Fig. 4.9 *Principales componentes del Modelo Navegacional*

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).

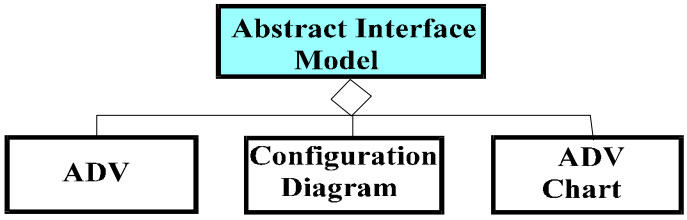
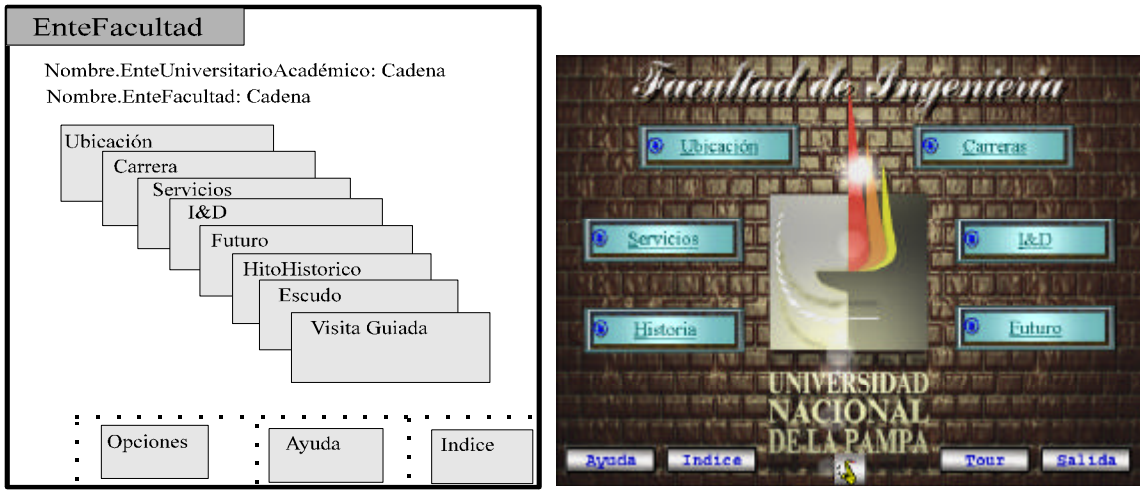


Fig. 4.11 *Diagrama de los constructores para el modelado de Interfaces Abstractas*

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.

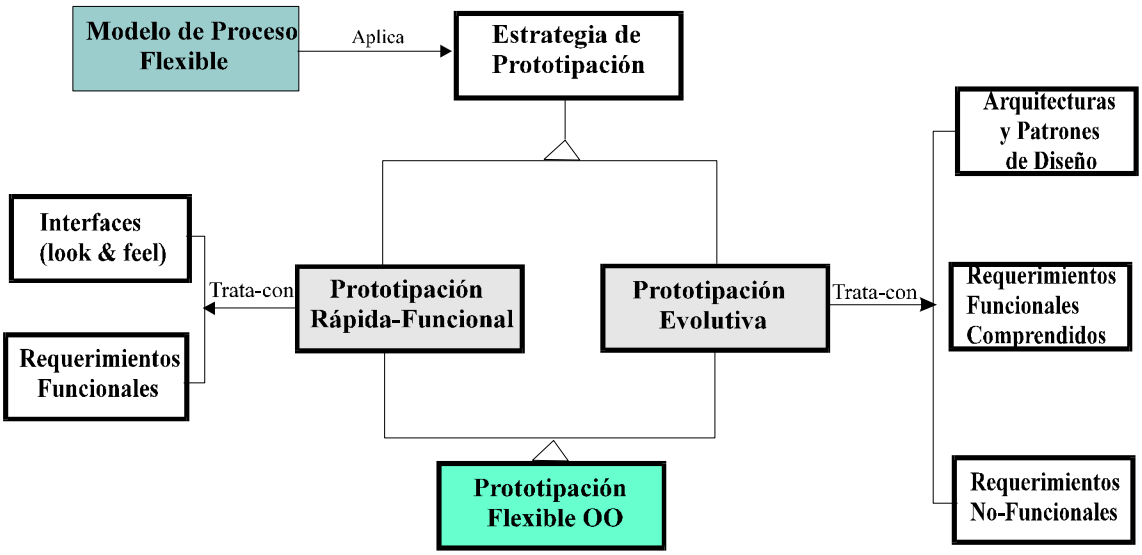


Fig. 4.13 *Diagrama de los principales responsabilidades de la estrategia de prototipación flexible.*

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 .

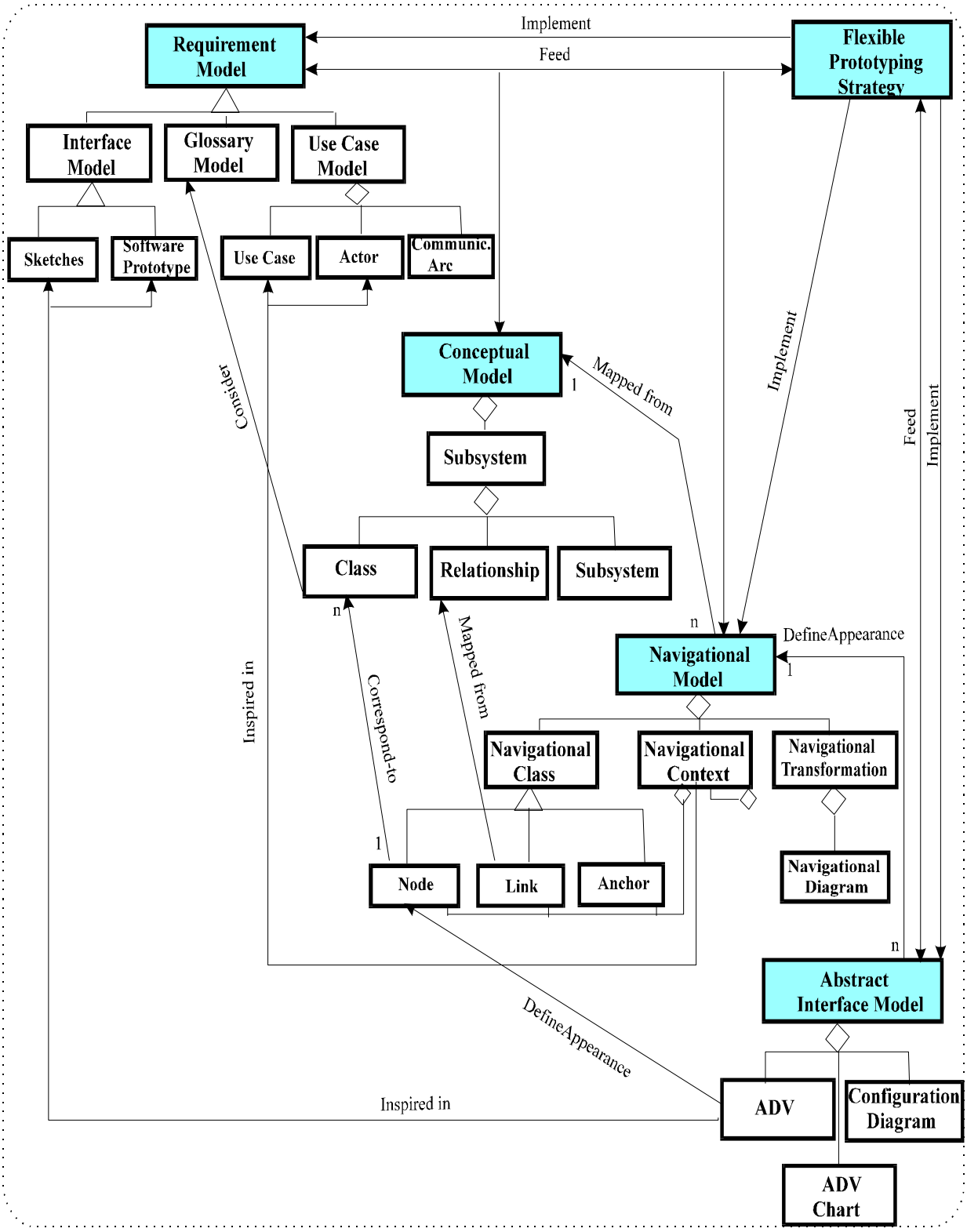


Fig. 4.14 *Diagrama que relaciona los modelos de requerimientos, conceptual, navegacional y de interfaces abstractas con la estrategia de prototipación flexible.*

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

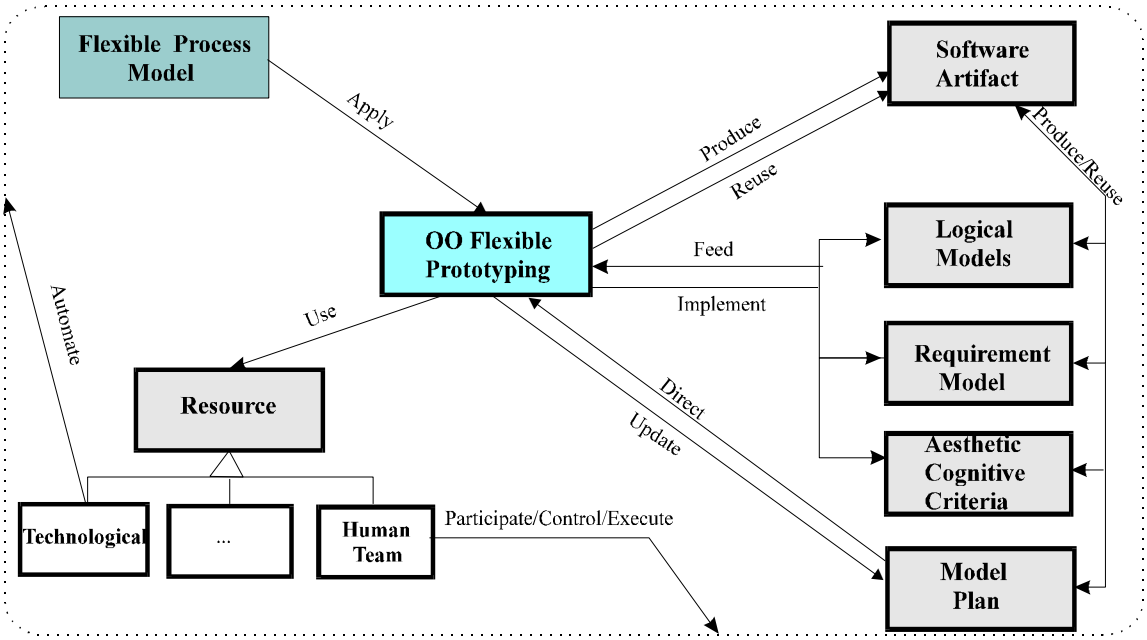


Fig. 4.15 *Diagrama general que relaciona la estrategia de prototipación flexible con algunos modelos lógicos y otras clases intervinientes.*

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.

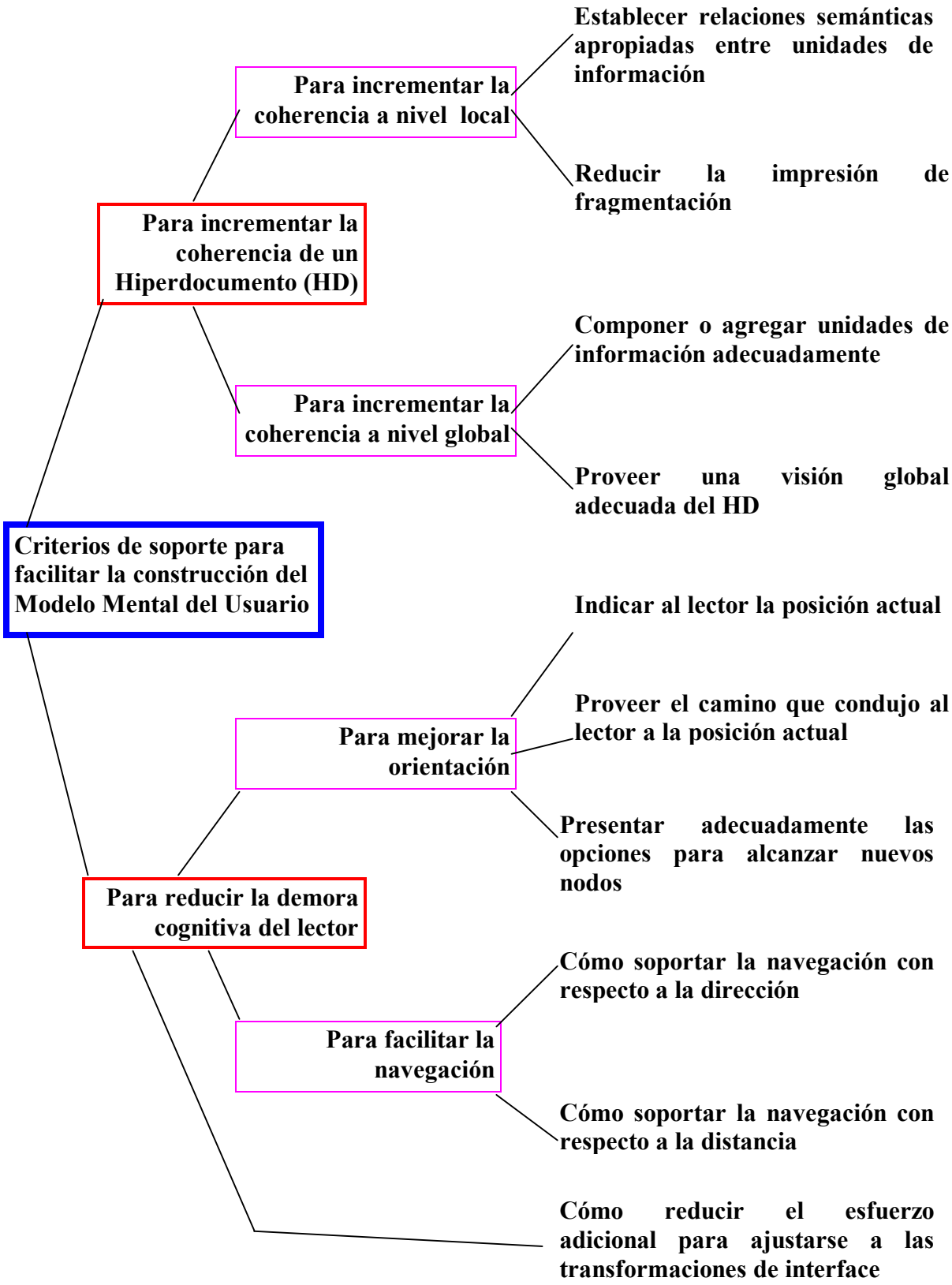


Figura 4.16: *Criterios Cognitivos en la creación de HiperDocumentos*

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).

5. Ingeniería de Hipermedia: mecanismos para la evaluación y control de Procesos, Artefactos y Recursos

Desafortunadamente en el campo de Hipermedia la mayoría de los artefactos se están produciendo por medio de una estrategia ad-hoc, no contemplando consecuentemente aspectos tradicionales de la Ingeniería de Software como administración de proyectos, modelado de procesos, establecimiento de métricas, mantenimiento de artefactos, por citar algunos. Sin embargo, como indicamos al final de la sección 2.1 en la comunidad científica se está comenzando a considerar la necesidad de contar con un enfoque amplio de proceso de desarrollo de hipermedia y, principalmente, se está viendo la necesidad de contar con un enfoque ingenieril: esto es, el empleo disciplinado, cuantificable y sistemático de principios de la Ingeniería de Software para la creación, evolución, evaluación y control de artefactos (y otros entes) en proyectos de hipermedia.

En este capítulo afirmamos que todo proceso de software debe velar continuamente por tres objetivos esenciales: desarrollar artefactos de calidad, utilizar los procesos óptimos y emplear los recursos apropiados. Lo anterior da pie para definir cuáles son los atributos y características observables que deben contribuir a la calidad de artefactos, procesos y recursos. Realizaremos una justificación de tal necesidad y mostraremos un listado de esas características en la sección 5.1. En la sección 5.2 presentaremos mecanismos de modelado de proceso que nos serán de utilidad para evaluar, analizar y controlar artefactos, procesos y recursos en el contexto de las metas establecidas de un proyecto de hipermedia. En este capítulo nos introduciremos en el tema de métricas (y heurísticas) en este nuevo campo.

5.1 Características deseables de Artefactos, Procesos y Recursos en Proyectos de Hipermedia

Una de las metas principales en el desarrollo de aplicaciones de hipermedia (como en cualquier otro tipo de desarrollo de software) es producir artefactos de calidad, los que deben estar regidos por un conjunto de atributos deseados y observables, utilizando para tal fin los procesos más óptimos y los recursos más apropiados para esas características. Debemos asegurar los mecanismos por medio de los cuales podemos construir artefactos de hipermedia que cumplimenten tales características a partir del planteo de un conjunto de requerimientos no funcionales.

Podemos ver al desarrollo de hipermedia como al proceso de producir artefactos los cuales contienen una equilibrada mezcla de atributos y características deseadas llevadas a cabo del modo más efectivo posible. Para que el desarrollo sea efectivo, en términos de la relación de compromiso entre calidad de los artefactos y costos de desarrollo, los procesos deben ser administrados de un manera óptima y eficiente.

Para los agentes intervinientes en el ciclo de desarrollo, los atributos representan una serie de restricciones a cumplir por los productos que se están construyendo, los procesos que se están empleando, y los recursos que se están asignando. Para que la descripción del modelo de proceso se complete es preciso definir los características y atributos que representen unidades de medida concreta para observar, retroalimentar, verificar, evaluar, analizar y predecir.

Podemos extraer de los conceptos expresados más arriba tres entes principales: artefactos, procesos y recursos. Por una parte, para comprender al proceso de desarrollo necesitamos comprender primeramente a los productos a ser desarrollados. Los atributos y la funcionalidad de los productos determinarán los atributos y la funcionalidad de los procesos. Para desarrollar procesos óptimos debemos considerar los artefactos a producir en el contexto de un tipo de proyecto y la relación entre actividades, productos y recursos disponibles. Por otra parte necesitamos considerar específicamente características de los procesos y de los recursos. Por ejemplo podemos tener procesos que produzcan un artefacto ideal (que cumplimente atributos de relevancia -de contenido y de enlaces, navegabilidad y nivel de cohesión) pero sea inviable o no factible desde el punto de vista de administración de tiempos y costos.

Lo anterior nos conduce a pensar que el objetivo básico (al menos en proyectos de mediana y gran escala) es tener procesos de calidad que nos aseguren una solución óptima y efectiva en cuanto a costos. De este modo, atributos de los productos, procesos, recursos (y hasta atributos de constructores de proceso), afectarán al proceso de desarrollo en diferentes formas. Para ilustrarlo con un ejemplo obvio pero conclusivo podemos decir que: en el armado de páginas de un sitio Web de una organización, el emplear los mejores expertos del mundo para tareas de diseño y autoría (ente "recurso humano" con atributos de extrema habilidad y experticia), podrá resultar en un sitio Web (artefacto) de muy alta calidad pero prohibitivo en cuanto a costos. De modo que para cualquier proceso de desarrollo se requiere una administrada relación de compromiso entre calidad y costos.

Una de las metas principales en el desarrollo de aplicaciones de hipermedia –que indicamos antes-, como en cualquier otro tipo de desarrollos es producir artefactos de calidad . Por lo tanto, ¿qué implica calidad en Hipermedia? Dado que cualquier desarrollo tiene un costo asociado, podemos considerar a un artefacto de calidad como a una aplicación, componente o documento que tiene una combinación óptima del conjunto de características y atributos deseados balanceados con el costo de cumplimentar con esos atributos. La calidad no es una medida absoluta sino relativa a un proyecto de hipermedia en el contexto de una organización y, en última instancia, al dominio de la aplicación.

La calidad de los artefactos producidos está relacionada con la calidad de los procesos de desarrollo utilizados. La interpretación de calidad como estar en conformancia con procesos estándar, ha llevado a prescribir plantillas de descripción de procesos, guías de estilo, etc. En la figura 5.1 mostramos a un script de proceso a un nivel medio de granularidad para la tarea T (de la figura 4.1).

Es importante tener en cuenta que los estándares pueden proveer una estructura y una guía para producir artefactos de calidad pero por sí solos no garantizan los resultados [Fenton 96]. Procesos de calidad combinados con recursos y constructores de proceso de calidad conducen con mayor probabilidad a resultados de calidad.

Task Class: **Cognitive Criteria Employment,** Alias: **T**
Goal: to assist the design of Web-based information spaces and their user interfaces by means of cognitive criteria so that it facilitates the quick comprehension of reading hyper-documents by the final user.
Super-tasks: Design Modeling, Authoring Criteria.
Subtasks: Coherence Criteria Employment, Orientation Criteria Employment, Navigation Criteria Employment, and Records of new Findings.
Inputs: Documented Cognitive Criteria, and Physical Models.
Outputs: Document about Coherence Criteria, Document about Orientation Criteria, Document about Navigation Criteria, and Records of new Findings.
Input Criteria: To be in the Development Phase
Output Criteria: Documentation about taken decisions or about new Findings
Comments: 1) One major proposes of reading hyper-documents by the final user is comprehension. In cognitive science, the readability of a document is characterized by the mental effort spent in building the mental model, which represents the objects and relationships of the hyper-document. Thüring et al propose a set of cognitive design issues or criteria that can be used to improve design, and respond to these questions: a) *How to increase local and global coherence*? b) *How to improve orientation*? c) *How to facilitate navigation*? d) *How to reduce additional effort for user-interface adjustment?* They identify a series of ten design issues, and from this, a series of eight principles.
2) They are independent of any methodology and should be prescribed by any hypermedia process model.

Figura 5.1 *Script de Descripción de Proceso para la Tarea* Cognitive Criteria Employment (extraído de [Olsina 98]).

Por último, para poder obtener artefactos de calidad se debe planificar, programar y controlar. Consecuentemente la calidad no podrá ser agregada a los artefactos de hipermedia al final del proceso de desarrollo sino que por el contrario se necesitará considerarla durante todo el ciclo de desarrollo; a la calidad se la debe planificar. La calidad es un resultado del proceso, y debe ser un factor directriz del mismo. Podremos direccionar el esfuerzo a realizar en ciertas tareas en una relación de compromiso con respecto a ciertos atributos (por ejemplo relevancia de contenido, navegabilidad y completitud) con el fin de mejorar la calidad de la aplicación final de hipermedia.

En la tabla 5.1 presentamos a un conjunto de características observables de los entes discutidos [Fenton et al 91, Goldberg et al 95] (que derivan en atributos mesurables -o potencialmente mesurables en algunos casos-, dado el carácter de novel del campo de hipermedia).

Desde el punto de vista de la evaluación, control y mejoramiento de procesos, artefactos y recursos, es preciso realizar mediciones sobre una o varias características. Las observaciones realizadas y los datos recolectados se pueden usar con propósitos de retroalimentación, valoración, predicción y control de las características de los entes. Por una parte, es oportuno recordar estos pares de máximas o principios:

1. "*No se puede medir lo que no se puede comprender*"
2. "*Si no se sabe dónde se está parado, un mapa no ayuda*"
3. "*Lo que no se puede medir no se puede controlar sistemáticamente*"
4. "*No podemos mejorar algo a menos que podamos medirlo"*

y, por otra parte, es importante recordar que una medida es un valor numérico computado a partir de un conjunto de datos observables y consistentes con la intuición, y para que tenga valor debe poseer las siguientes características:

- La medida debe ser robusta: el cálculo es repetible e insensible a pequeños cambios en el entorno, herramientas y observadores.
- La recolección de los datos debe soportar los principios de objetividad científica.
- La medida debe establecer escalas y límites
- La medida debe ser relevante respecto al proceso, producto o recurso
- Los datos deben ser fáciles de recolectar y validar

Tabla 5.1 *Características observables, internas y externas de los entes Artefacto, Proceso y Recurso*

Atributo / Ente	Interno (Objetivo)	Externo (Subjetivo)
Artefacto	• Relevancia (de contenido, de enlaces) • Nivel de cohesión o fragmentación, organización de conceptos (a nivel local – agrupamiento de los componentes (atributos, nodo o página) en un contexto de navegación, y a nivel global –entre contextos o aplicaciones diferentes) • Completitud • Acoplamiento (entre contextos) • Reusabilidad • Cantidad de defectos (por ej. Nodos destino ausentes, enlaces a tipo de nodos inválidos, etc.) , frecuencia • Tamaño (cantidad de nodos, cantidad de enlaces por nodo, cantidad de atributos por nodo) • Complejidad (del grafo de nodos, centralidad, nivel de interconexión,etc.) • Nivel de Documentación	• Navegabilidad (nivel de interconexión, distancia, orientación, camino, etc.) • Confiabilidad (alcanzabilidad de los nodos destino, validez de los enlaces, etc.) • Mantenibilidad • Usabilidad (autoevidencia de los objetos presentados, mecanismos de búsqueda, mecanismos de índices, anotaciones, interfaces, etc.). • Seguridad • Calidad • Nivel de Satisfacción • Legibilidad (del hiperdocumento)
Proceso	• Flexibilidad (para incorporar descripciones de proceso alternativas, instanciable en varios proyectos), escalabilidad • Tiempo • Completitud • Esfuerzo • Relevancia (de contenido) • Reusabilidad • Cantidad de defectos, frecuencia • Repetitividad	• Costo • Calidad • Estabilidad • Mantenibilidad • Comprensibilidad • Performance
Recurso	• Habilidad (de un agente humano) • Años de Experiencia • Tamaño (de un equipo de trabajo) • Nivel de Comunicación • Nivel de Estructuración • Velocidad (en hardware, etc.) • Capacidad de Memoria • Temperatura (en una oficina) • Luz	• Costo (por ej. precio de un agente humano, en función de la habilidad y años de experiencia) • Productividad • Calidad • Usabilidad • Nivel de Confort

Desafortunadamente en el área de hipermedia hasta el presente, hay muy pocas métricas investigadas (por ej. las más formalizadas son las contenidas en [Botafogo et al. 92] que han realizado estudios sobre métricas estructurales, respecto a complejidad, de grafos de nodos y enlaces), y, de las métricas existentes en hipermedia, falta la experiencia suficiente para interpretarlas de manera que sean relevantes, efectivas, al mismo tiempo que fáciles de recolectar y validar (considere por ejemplo las métricas de relevancia de contenido y enlaces, siendo ésta característica tan importante en aplicaciones de soporte a la educación y aprendizaje).

En la tabla anterior hemos seguido algunos criterios existentes en cuanto a la categorización de las métricas, y hemos dividido a las características o atributos de los entes, en internos (objetivos) y externos (subjetivos, es decir, características de los entes relacionados a un sujeto o agente). Algunas de las características de la tabla requieren mayor investigación (que no es el objetivo de esta tesis profundizar en ellas).

En la siguiente sección veremos mecanismos para seleccionar métricas en función de metas y objetivos particularmente el enfoque GQM (Goal-Question-Metric approach) [Basili et al. 84, 94]. Dado un conjunto de metas del proyecto (con ciertas características, seleccionadas de la tabla), debemos realizar un conjunto de preguntas relevantes y, en función de esto elegir las métricas que mejor interpreten a las preguntas. Los resultados deben servir para mejorar procesos (artefactos, recursos), para analizar, predecir, controlar.
Con objetivos similares podemos utilizar a las Plantillas de Calidad (Quality Templates) [Gilb 88] como una herramienta necesaria para especificar atributos mesurables (y los prerequisitos previos a la medición) para un proyecto específico.

Sin embargo, debemos contar primero con una estructura conceptual, que integre enfoques descriptivos y prescriptivos de modelado de procesos en el cual el modelado se deba llevar a cabo (para algunos o todos de los entes representados en el modelo conceptual, discutido en el capítulo 3). Este tipo de estructura conceptual, particularmente en el campo de Hipermedia, es la que estamos construyendo en base a la experiencia previa de la Ingeniería de Software, que es muy amplia, y en base a la experiencia más reciente en este nuevo campo, con sus metodologías, técnicas y heurísticas.

5.2 Enfoque de Modelado de Procesos para evaluar, analizar y controlar Artefactos, Procesos y Recursos

5.2.1 Estructura integradora para enfoques descriptivos y prescriptivos.

Como indicamos previamente uno de los objetivos básicos del Modelo de Proceso Flexible de Hipermedia es tratar de prescribir un conjunto de submodelos y, con respecto a la vista funcional, prescribir a un conjunto de tareas para las distintas fases. En el capítulo 4 nos centramos principalmente en la fase de desarrollo, representando tareas con el propósito de comunicar y comprender aspectos del proceso. No obstante el modelo de proceso debe servir también al objetivo de mejoramiento de los procesos.

Para lograr ambos objetivos, vemos a la tarea de diseñar a un modelo de proceso como un espacio de trabajo bidimensional, con espacios de retroalimentación, esto es, observar y estudiar procesos de desarrollo de hipermedia existentes y abstraer y prescribir procesos deseados. En la figura 5.2 representamos una estructura conceptual de enfoques descriptivos y prescriptivos para el modelado de procesos.

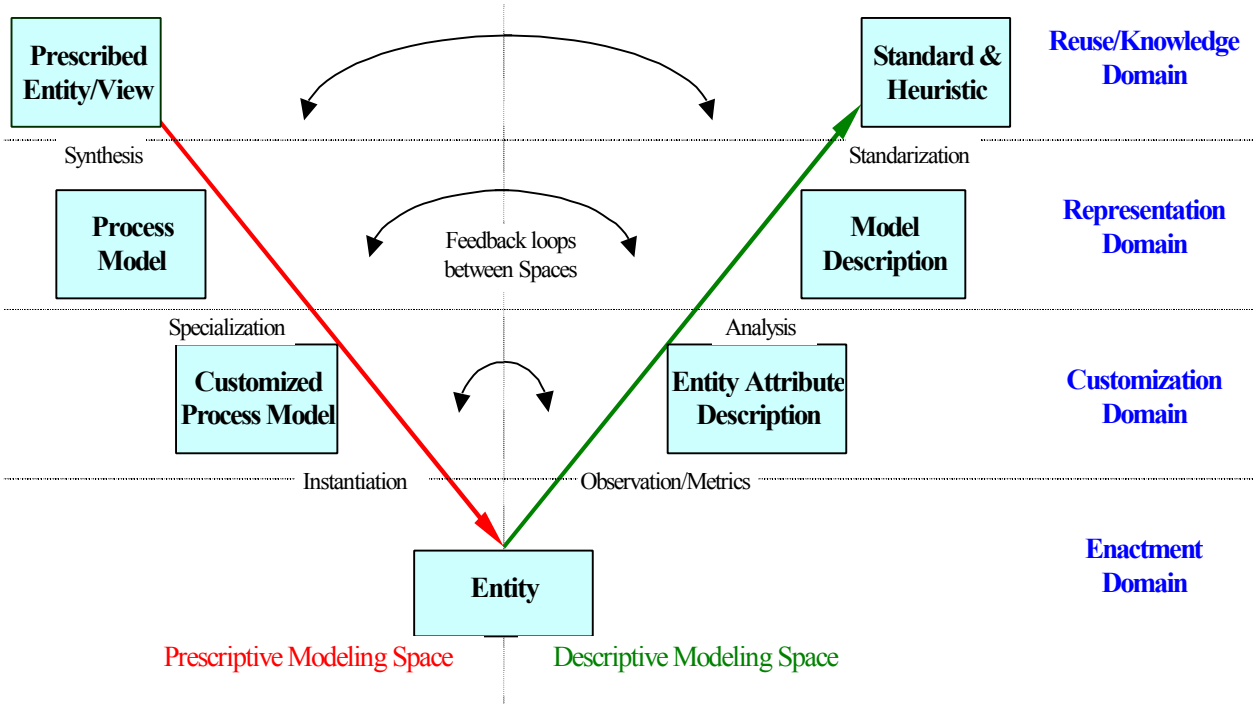


Figura 5.2 *Relación entre espacios y dominios de modelado descriptivo y prescriptivo.*

Para explicar en principio esta relación entre espacios y dominios de modelado prescriptivo y descriptivo nos será de utilidad los conceptos y terminología usada por Lonchamps (que transcribimos en su lenguaje original).

"*People dealing with software processes may adopt two different attitudes of mind:*

- Descriptive: *they study existing processes to answer the question "how software is (or has been) actually developed?"*
- Prescriptive: *they define desired processes to answer the question "how software should be developed?"..."*

En ambas actitudes, prescriptiva y descriptiva, los ingenieros de procesos deben apuntar a:

- "Expressing: *the actual or desired process is just described more or less formally for understanding, communication, education, reuse, or standardization.*
- Analyzing: *the description of the actual or desired process is studied through more or less formal techniques (such as validation, e.g. simulation, or property verification) for a deeper understanding, comparisons, improvement, impact analysis, or forecasting.* " [Lonchamps 93].

5.2.2 Mecanismos de modelado de procesos para enfoques descriptivos.

Como indicamos previamente, las datos recolectados de las mediciones una vez interpretados son útiles para valorar, predecir, y controlar alguna de las características de los artefactos, procesos o recursos. A partir de la interpretación y análisis de los datos las métricas proveen de un modo sistemático, el aprendizaje de la experiencia pasada o reciente para aplicarlo a los procesos, artefactos y recursos de un proyecto actual o a situaciones futuras.

Ahora la pregunta es, a partir de qué enfoque o estrategia seleccionamos las características observables (relevancia de enlaces y contenido, navegabilidad, etc.) y en qué contexto las analizamos e interpretamos? De acuerdo a investigaciones realizadas las métricas para que sean efectivas deben estar focalizadas hacia metas específicas, aplicadas a todo el ciclo de vida de los entes e interpretadas en función de la comprensión del contexto organizacional [Basili et al. 94]. Esto implica que las mediciones se deben definir bajo una estrategia de arriba hacia abajo (top-down).

El enfoque GQM (Goal-Question-Metric approach) responde a esa estrategia, esto es, deriva mediciones a partir de metas (goals). Dado un conjunto seleccionado de metas del proyecto en el contexto de una organización (teniendo en cuenta las características y atributos deseables de los artefactos, productos y recursos), se construye y refina un conjunto de preguntas (questions) para cada meta y, en función de cada pregunta se eligen las métricas (metrics)

apropiadas. Los resultados de las interpretaciones pueden servir para analizar, comprender y mejorar procesos, productos y recursos (para alimentar principalmente al enfoque descriptivo de modelado de procesos), como así también con propósitos de ganar control en la administración de proyectos y con propósitos de predicción.

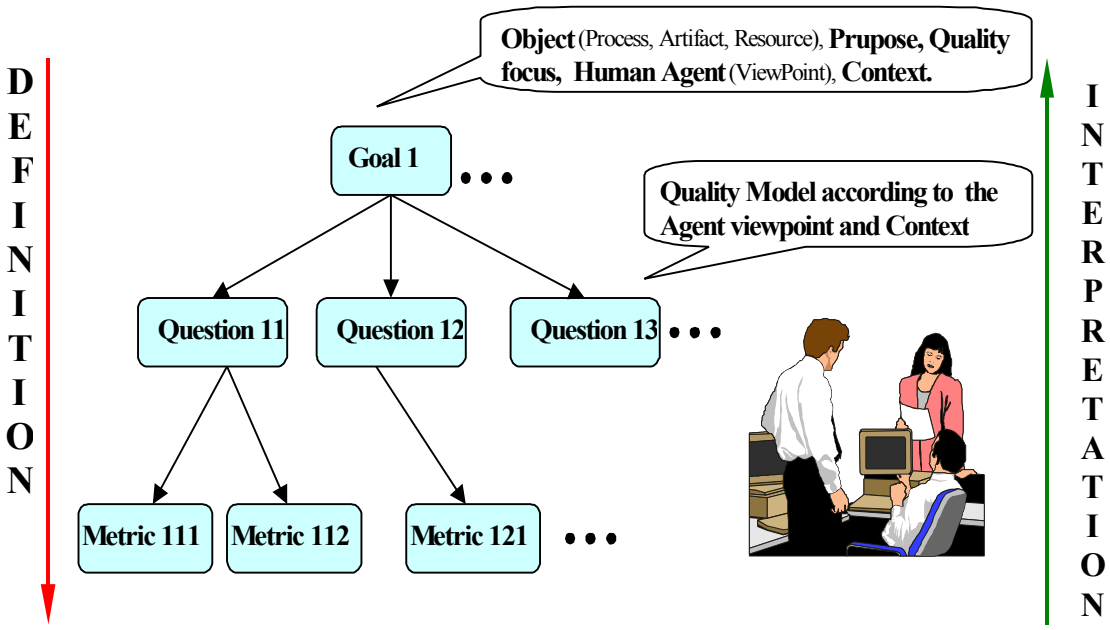


Figura 5.3 *Estructura Jerárquica del Enfoque Goal-Question-Metric. Las flechas representan dos estadios relevantes del proceso: el de definición y el de interpretación.*

En la figura 5.3 se pueden apreciar los componentes del modelo conformando una estructura jerárquica. A seguir describimos de un modo informativo, los tres niveles del modelo GQM [Basili et al. 94]:

- ✓ *Nivel Conceptual* (Meta): una meta se define para un ente u objeto, para uno o varios propósitos u objetivos, con respecto a modelos de calidad (características), para algún agente humano en cumplimiento de algún rol (punto de vista), en algún contexto particular.
- ✓ *Nivel Operativo* (Pregunta): se refina un conjunto de preguntas a partir de una meta, identificando el objeto de medición con respecto a características de calidad seleccionadas para un punto de vista.
- ✓ *Nivel Cuantitativo* (Métrica): se refina un conjunto de métricas para cada pregunta, de modo de responder a cada una de ellas de un modo cuantitativo (conjunto de datos recolectados a partir de atributos observables -atributos internos u objetivos, externos o subjetivos).

Para ilustrar con un ejemplo el empleo de este enfoque y teniendo en cuenta una meta propia del campo de Hipermedia, supongamos que la misma consiste en "*Mejorar la navegabilidad de un hiperdocumento desde el punto de vista del usuario final*" se pueden formular preguntas y a partir de éstas refinar métricas. En la tabla 5.2 presentamos una plantilla para capturar esta información conforme al esquema antes discutido.

Tabla 5.2 *Plantilla para registrar Metas, Preguntas, Métricas y Comentarios*

Meta 1	
Propósito u Objetivo	*Mejorar*
Característica o Atributo	*Navegabilidad*
Objeto (tipo)	*Hiperdocumento (artefacto)*
Agente asignado a un rol	*Usuario final*
Pregunta	
P11	*Cuál es el nivel de interconectividad apropiado entre nodos pertenecientes a un grafo de hipermedia?*
Métrica	
Me111	*Nivel de Interconexión (NI) = (Max - Suma) / (Max-Min)* *Max = $(n^2-n) * K$* *n= cant. de nodos del grafo* *K=constante superior a la cantidad de nodos* *Min= (n^2-n)* *Suma representa a la suma total de distancias tomadas a partir de la matriz de distancias convertidas (con el factor K)* *Suma= $_i \ _k \ D_{ik}$ en donde D_{ik} representa la distancia entre los nodos ik*
Me112	*Evaluación subjetiva del prototipo (en revisión conjunta con el usuario final) para validar interconexión entre nodos.*
Pregunta	
P12	*Cuál es el nivel de alcanzabilidad óptima entre dos nodos?*
Métrica	
Me121	*Distancia mínima (dado que pueden existir caminos alternativos para alcanzar dos nodos, se requiere verificar todos los caminos que permitan navegación entre los mismos)*
Me122	*Distancia mínima promedio*
Me123	*Desviación estándar (dada la distancia mínima promedio con respecto a cierto umbral)*

Comentarios	
M1.1	*El objetivo de **Mejorar** está inserto en el contexto de un modelo de calidad a seguir para el proyecto y organización.*
M1.2	*Respecto a la característica de **Navegabilidad** se puede ver afectada por aspectos tales como el grado de conexión entre nodos de un contexto navegacional; la existencia de caminos apropiados entre dos nodos intervinientes (por ej. la distancia más corta a partir de una deseada) y por la facilidad de alcanzarlos (orientación, ajuste de interface).*
M1.3	***Hiperdocumento** Es un documento con propiedades de hipertexto (o hipermedia)*
Me111.1	***NI** se refiere al grado de interconexión de nodos de un grafo (o estructura) de hipermedia. Es un valor entre 0 y 1 en donde cero implica que no hay conexiones entre nodos y uno implica conexión total. La experiencia y el análisis recomienda como apropiado (según Botafogo et al. 92) un rango entre 0.3 y 0.8.*
Me112.1	*Demasiados anchors en los nodos (y sus enlaces asociados a nodos destinos) puede indicar un hiperdocumento pobremente organizado.*

5.2.3 Comentarios relacionados.

Otro aspecto preponderante en el enfoque GQM es la interpretación de los datos capturados en función de las preguntas a partir de las cuales se derivaron esas medidas. Por ejemplo para la métrica M111 la podemos interpretar como una valoración de la complejidad de enlaces (ponderada) entre nodos de un grafo. Desde el punto de vista del usuario una muy alta interconexión indicará que cada nodo tiene muchos puntos de partida y enlaces a nodos intermedios o destinos. Esto puede atentar contra la calidad del producto (en el atributo de navegabilidad) al permitir elegir potencialmente al mismo tiempo diversos destinos, pudiendo desorientar al usuario. En un hiperdocumento totalmente interconectado el usuario no tiene un camino bien estructurado para seleccionar un artículo o completar un concepto distribuído en varios nodos. El caso opuesto también es generalmente un indicio de mal diseño.

En la introducción de este capítulo afirmamos que todo proceso de desarrollo de software debe

velar continuamente dado un contexto por tres objetivos esenciales: construir artefactos de calidad, utilizar los procesos óptimos y emplear los recursos apropiados. Esto nos permite definir cuáles son los características y atributos observables que deben contribuir a la calidad de artefactos, procesos y recursos. Mediante el enfoque GQM (y otras herramientas más específicas como las Plantillas de Calidad), podemos definir y planificar situaciones deseadas, que por medio de la recolección de datos relevantes nos permitan evaluar, controlar y en definitiva mejorar procesos, artefactos y recursos.

Finalmente identificaremos a un conjunto de metas que pueden ser consideradas en proyectos de hipermedia (formulando sólo el nivel conceptual del modelo GQM). La siguiente lista no pretende ser de ningún modo extensiva:

- ✓ *Evaluar la relevancia y completitud del contenido de un hiperdocumento* **-Propósito:** *Evaluar* **Característica o Atributo:** *Relevancia y Completitud de contenidos* **Objeto (tipo):** *Hiperdocumento (artefacto)*
- ✓ *Reducir costos durante las tareas de requerimientos, diseño y autoría* **-Propósito:** *Reducir* **Característica o Atributo:** *Costo* **Objeto (tipo):** *Tareas de requerimientos, diseño y autoría (proceso)* **Comentario:** *Esta meta se puede refinar en principio en tres submetas, relacionadas a los tres procesos*
- ✓ *Mejorar el personal en el diseño gráfico de páginas Web* **-Propósito:** *Mejorar* **Característica o Atributo:** *Habilidad (diseño gráfico en la Web)* **Objeto (tipo):** *Personal (recurso)*
- ✓ *Maximizar el reuso de componentes en la fase de desarrollo* **-Propósito:** *Maximizar* **Característica o Atributo:** *Reusabilidad* **Objeto (tipo):** *Componentes (artefacto)*
- ✓ *Mejorar la confiabilidad en la navegación del hiperdocumento* **-Propósito:** *Mejorar* **Característica o Atributo:** *Confiabilidad* **Objeto (tipo):** *Hiperdocumento (artefacto)*
- ✓ *Mejorar la performance en actividades de autoría para generar enlaces apropiados* **-Propósito:** *Mejorar* **Característica o Atributo:** *Performance* **Objeto (tipo):** *Actividades de autoría (proceso)*

5.3 Aspectos relacionados al Proyecto "*Facultad de Ingeniería*"

En la sección 2.4.1 presentamos el trabajo de autoría utilizado en esta tesis para ejemplificar distintos aspectos, el cual se construyó con el modelo de proceso de hipermedia. La aplicación representa una vista del modelo conceptual de un SIA (fig. 4.9) y el perfil del usuario considerado es el del estudiante.

Primero determinamos el alcance del trabajo de autoría a producir y capturamos preliminarmente (en una semana y media) un conjunto de requerimientos iniciales. Se analizó la factibilidad de introducir la aplicación final en un medio de almacenamiento masivo como CD-ROM (año 1995). Las decisiones de planificación consistieron en fijar estrategias y objetivos de desarrollo, en elegir los constructores de proceso y herramientas, el tipo de usuarios que participarían en las demostraciones (en las primeras iteraciones del PRF participaron dos estudiantes y una autoridad académica; y en las demostraciones del prototipo evolutivo participaron grupos de cinco personas), y, entre otros aspectos, el modelo de calidad a seguir.

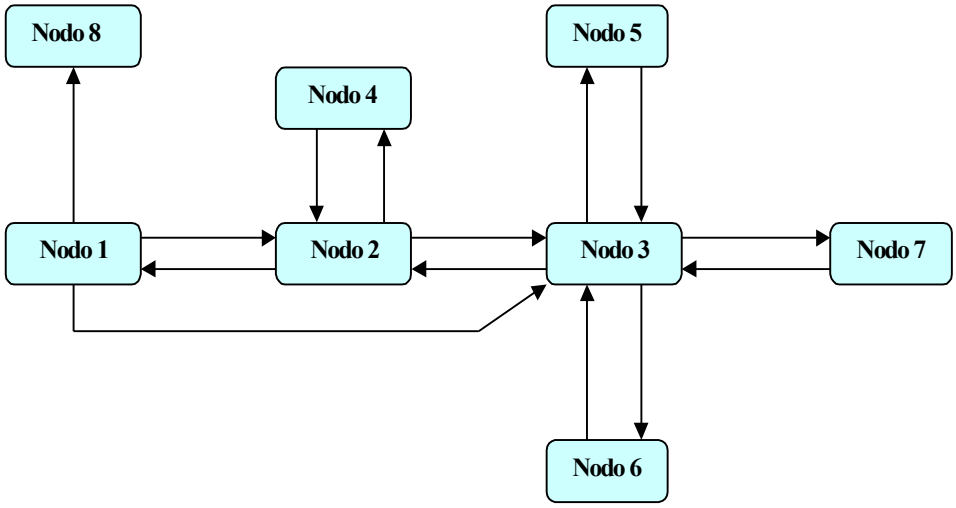


Figura 5.4 *Estructura o grafo de nodos del contexto "Ubicación"*

A la segunda semana comenzamos a prototipar la presentación animada y la clase *"EnteFacultad"* (figs. 2.2 y 2.3a); paralelamente comenzamos a estudiar dos contextos navegacionales "*Ubicación*" y "*Carreras*" distribuyendo el trabajo en dos subgrupos. El total de personas integrantes del equipo fueron cuatro (no considerando en este número a los usuarios participantes).

En cuanto a los atributos de calidad planteados dentro de los requerimientos no funcionales, se encontraban la usabilidad (mecanismos de índices, visitas guiadas, recorrido histórico, ajuste de interface) y aspectos de navegabilidad, entre otras características.

Para ilustrar con un ejemplo el empleo del enfoque GQM tratado en la sección anterior y, teniendo en cuenta que nos propusimos como meta mejorar la navegabilidad de la aplicación desde la perspectiva del estudiante, nos formulamos preguntas para luego observar y analizar el nivel de navegabilidad de los contextos de navegación. Por ejemplo, previo al desarrollo del prototipo del contexto "*Ubicación*" especificamos una estructura de navegación tentativa y mecanismos de recorrido de los nodos. El grafo resultante es el mostrado en la figura 5.4.

a) b)

Figura 5.5 a) *Implementación del nodo 1 -ver fig. 5.4- del contexto "Ubicación"* ; **b)** *Implementación del nodo 2*

a) b)

Figura 5.6 a) *Implementación del nodo 3 -ver fig. 5.4*; **b)** *Implementación del nodo 8, correspondiente al primer nodo del contexto "Carreras"*

Si observamos los nodos y enlaces del grafo de la figura 5.4, encontraremos varias de las correspondencias con objetos de las pantallas mostradas en las figuras 5.5 y 5.6. Los botones "*La Pampa*" y "*Gral. Pico*" de la fig. 5.5a son los anchors que conducen a los nodos 2 (fig. 5.5.b) y 3 (fig. 5.6.a) y el texto en rojo (o anchor) de la misma figura conduce al nodo 8 (fig 5.6.b). Asimismo podemos apreciar los controles de navegación direccionales "anterior" y

"siguiente", que se corresponden al doble enlace entre nodos del grafo.

En la tabla 5.3 presentamos una plantilla para volcar preguntas y métricas conforme a la meta planteada: "*Mejorar la navegabilidad de la aplicación Facultad de Ingeniería desde la perspectiva del estudiante*".

Tabla 5.3 *Plantilla para registrar Metas, Preguntas y Métricas específicas al proyecto Facultad de Ingeniería*

Meta 1 **Propósito u Objetivo** **Característica o Atributo** **Objeto (tipo)** **Agente asignado a un rol**	 *Mejorar* *Navegabilidad* *Aplicación Facultad de Ingeniería (artefacto)* *Estudiante*
Pregunta **P11**	*Cuál es el nivel de interconectividad entre nodos pertenecientes a cada contexto de navegación interviniente?*
Métrica **Me111**	*Nivel de Interconexión (NI) = (Max - Suma) / (Max-Min)*
Me112	*Validación subjetiva del prototipo en revisión conjunta con el usuario final (estudiante) para evaluar el nivel de interconexión entre nodos (para cada contexto navegacional).*
Pregunta **P12**	*Cuál es el nivel de alcanzabilidad óptima entre dos nodos no superior a un umbral?*
Métrica **Me121**	*Distancia entre dos nodos no superior a cuatro saltos*
Me122	*Distancia mínima promedio*
Comentarios **M111.1**	✓ *El valor determinado para el grafo correspondiente al contexto "Ubicación" es de 0,75 (ver fig. 5.4).* ✓ *El valor determinado para el grafo correspondiente al contexto "Carrera" es de ...*

Para la formulación de la pregunta P11 encontramos dos métricas de cuyo análisis (de los valores recolectados, principalmente para la métrica M111), nos permite realizar una

evaluación de los objetivos fijados con mayor rigor científico. La métrica M112 consistió en una validación del prototipo en las sesiones de demostración previstas para evaluar el nivel de interconexión entre nodos (para cada contexto navegacional). Además las sesiones se utilizaron para evaluar la relevancia de contenidos y enlaces, y la usabilidad del prototipo.

En este trabajo presentamos en la figura 5.7 el valor determinado de nivel de interconexión de nodos del contexto "*Ubicación*" conforme al grafo visto (esto se puede hacer extensivo a los contextos de navegación restantes).

A Desde	*Nodo 1*	*Nodo 2*	*Nodo 3*	*Nodo 4*	*Nodo 5*	*Nodo 6*	*Nodo 7*	*Nodo 8*	*Suma Fila*	*Centralidad*
Nodo 1	0	1	1	2	2	2	2	1	11	0,067
Nodo 2	1	0	1	1	2	2	2	2	11	0,067
Nodo 3	2	1	0	2	1	1	1	3	11	0,067
Nodo 4	2	1	2	0	3	3	3	3	17	0,103
Nodo 5	3	2	1	3	0	2	2	4	17	0,103
Nodo 6	3	2	1	3	2	0	2	4	17	0,103
Nodo 7	3	2	1	3	2	2	0	4	17	0,103
Nodo 8	9	9	9	9	9	9	9	0	63	0,384
								Suma Total	**164**	

K = 9; n = 8

Max = $(n^2-n) * K = 504$

Min = $(n^2-n) = 56$

NI = (Max – Suma) / (Max – Min) = 0,75

Figura 5.7 *Matriz de Distancias Convertidas entre nodos del grafo de la fig. 5.4 y valor calculado de Centralidad y Nivel de Interconexión*

De la interpretación de estos datos y de las demostraciones conjuntas efectuadas con los usuarios finales, se determinó que el nivel de interconectividad es el adecuado para este tipo aplicación. (Recordar que NI es un valor que va en un rango entre 0 y 1 en donde cero implica que no hay conexión entre nodos y uno implica conexión total. La experiencia y el análisis recomienda como apropiado un rango entre 0.3 y 0.8, dependiendo del tipo de aplicación de hipermedia).

6. Conclusiones y Líneas de Trabajo Futuras

Como indicamos en la introducción, el advenimiento de los CD-ROM de multimedia y de la Web, desde inicios de la década del 90, han marcado un rápido crecimiento en los desarrollos de hipermedia. Este crecimiento de aplicaciones, principalmente de mediana y gran envergadura, no ha sido acompañado por un modelo de ciclo de vida bien definido que favorezca estrategias de planificación y control, de reuso y de mejoramiento de artefactos, procesos y recursos. Comentamos que la estrategia hasta ahora utilizada ha sido mas bien una circunstancial, o también citada en la literatura como estrategia "ad hoc" o "just-do-it". Pensamos que un uso más sistemático y riguroso de principios, modelos y métodos de Ingeniería de Software para la construcción de artefactos de hipermedia puede contribuir esencialmente a mejorar distintos atributos del proceso de desarrollo y de los productos generados.

Un proceso de desarrollo de hipermedia (y cualquier proceso de software en general), puede proceder en varias direcciones y etapas simultáneamente, respetando algún orden parcial en la realización de las tareas. Un modelo de proceso de hipermedia puede ofrecer guías en todo momento para realizar mejores elecciones, pero no debe prescribir reglas rígidas que sean válidas en todas las circunstancias. Sin embargo, pensamos que el empleo sistemático de un modelo de ciclo de vida puede contribuir favorablemente en la disminución del fenómeno denominado crisis del software.

En esta tesis hemos propuesto un nuevo enfoque para representar a los procesos de software y para asistir en la creación, evolución y mantenimiento de artefactos de hipermedia, denominado Modelo de Proceso Flexible. El mismo abarca fases de exploración, desarrollo y de vida útil de los productos. Hemos delineado sus perspectivas e introducido aspectos de algunas de estas vistas. Presentamos, en un nivel de granularidad media, las fases, tareas y actividades del modelo, concentrándonos principalmente en la fase de desarrollo. Entre algunos beneficios que deseamos apuntar se encuentran:

- Los modelos construidos están basados o soportan el paradigma Orientado a Objetos, el cual favorece al proceso de desarrollo por medio de mecanismos como abstracción, generalización/especialización, encapsulamiento, agregación, etc., y contribuye al proceso de comunicación entre los participantes al permitir un lenguaje común.
- Los constructores de proceso de las distintas tareas, al estar basados en modelos, permiten encarar proyectos de mayor complejidad.
- Se establece una clara división de preocupaciones entre los distintos procesos del modelo cubriendo todas las fases y tareas esenciales de un proyecto de hipermedia.
- Esta clara división produce mayor visibilidad al proyecto la cual, en definitiva, puede contribuir a la planificación, programación, control y establecimiento de métricas.
- El uso sistemático de modelado lógico y físico como el propuesto favorece al proceso de especificación y construcción en un contexto de desarrollo participativo.
- Ofrece constructores arquitectónicos de alto nivel como por ejemplo, casos de uso, contextos navegacionales de OOHDM, etc.
- Propicia la estandarización y el mejoramiento de los procesos.
- Permite definir un modelo de seguimiento de artefactos de software. Esto favorece esencialmente al desarrollo y mantenimiento de productos en proyectos de mediana y gran envergadura.

Por otra parte, definimos y representamos una estructura conceptual para el dominio de modelado de procesos la cual pretende ser el repositorio básico de los objetos y relaciones principales intervientes en todo modelo de proceso general. La definición de estos entes favorece a la modelización de las distintas perspectivas propuestas, a saber: funcional, informacional, de comportamiento, metodológica y organizacional. Una división de preocupaciones por perspectivas puede disminuir la complejidad en el modelado de procesos. De este modo, se puede separar diferentes tipos de información de los procesos para especificar, comunicar y controlar porciones del meta-modelo.

Pensamos que la definición de un marco conceptual base potencia a las investigaciones en el área, al permitir:

- la posibilidad de combinar diferentes clases y relaciones para dar soporte a las distintas perspectivas.
- la posibilidad de combinar formalismos y/o notaciones para describir a los procesos que, en definitiva, propenderá a la estandarización de los mismos.
- la posibilidad de crear capacidades en ambientes de ingeniería de software centrado en procesos para que soporten multi-paradigmas (tanto en el proceso de guía como de ejecución de actividades).

Finalmente, argumentamos respecto de la necesidad de desarrollar artefactos de hipermedia de calidad, de utilizar los procesos óptimos y de seleccionar los recursos apropiados en el contexto de las metas establecidas de un proyecto de hipermedia. Definimos una lista de atributos y características que pueden contribuir a un modelo de calidad y presentaremos enfoques de modelado de proceso útiles para analizar y mejorar procesos y productos. Si bien introdujimos el tema de métricas para el área de hipermedia este punto solamente merecería ser argumento para otra tesis de Master.

6. 1 Futuros Avances

La disciplina de Ingeniería de Hipermedia es un campo bastante reciente, fértil y extremadamente dinámico. Para ser sinceros, muchas son las direcciones de investigación ya abiertas o que se pueden abrir en el área de procesos de desarrollo de artefactos de hipermedia y modelización de procesos. Podemos citar los siguientes futuros avances de algunos temas no tratados en la presente tesis u otros inicial o parcialmente desarrollados:

Perspectivas de un Modelo de Proceso : propusimos y desarrollamos algunos aspectos de las perspectivas funcional, informacional, de comportamiento, organizacional y metodológica.
- Para la perspectiva funcional debemos todavía especificar a un más bajo nivel de granularidad, tareas y actividades y relacionarlas con roles y habilidades de los agentes. Un esquema interesante de descripción de tareas en función de roles y de otros atributos se presenta en Goldberg et al 95.

- Para la perspectiva informacional hay potenciales líneas abiertas en lo que se refiere a modelos de seguimiento de artefactos y configuración de cambios.
- En cuanto a la perspectiva de comportamiento sólo hemos indicado de un modo descriptivo e inicial la manera en que se realizan las tareas cuando se utiliza una composición de estrategias, principalmente comentada para la fase de desarrollo. Habría que estudiar mecanismos de concurrencia y sincronización e intentar utilizar formalismos para especificar la dinámica de los procesos (para esto puede ser útil redes de Petri, statecharts, etc).
- Para la perspectiva metodológica es quizás donde más avances se han realizado dentro del campo de Hipermedia para el desarrollo de aplicaciones en la Web y de autoría, principalmente para tareas de diseño y construcción, como hemos discutido en la sección 2.2 y en el capítulo 4.

Lenguajes de descripción de procesos y ambientes de ingeniería de software centrados en procesos: un área reciente de investigación consiste en el diseño y construcción de entornos centrados en procesos que ofrezcan asistencia en la guía y ejecución de procesos de software. Luego del análisis de la literatura investigada, y de las observaciones iniciales efectuadas en los proyectos de desarrollo de hipermedia en que hemos participado, se puede indicar que:
- No existe una única notación universal para especificar a todos los aspectos de un modelo de proceso. Como consecuencia se requiere un lenguaje multi-paradigma para describir y especificar a los distintos objetos y relaciones. El ambiente debiera soportar los diferentes tipos de descripción de procesos (formales, semiformales e informales), para ser ejecutados por agentes humanos o automáticos.
- Dada la complejidad de los elementos y fenómenos que intervienen en un proyecto de software, es importante centrar la atención en las perspectivas del modelo de proceso para ofrecer facilidades de ambiente.
- Un ambiente debería proveer capacidad de redefinición dinámica de descripciones para soportar aspectos evolutivos del proceso de desarrollo.

Glosario

Aclaraciones

Los conceptos del glosario están ordenados alfabéticamente. Un concepto puede ser un único término como **meta** o una frase como **ambiente de ingeniería de software centrado en procesos.**

Utilizaremos la/s palabra/s resaltada/s en **negrita** para describir un nuevo concepto. Emplearemos un estilo *itálico* para indicar que dicho término o frase se referencia en el glosario. Por último usaremos paréntesis para referirnos a sinónimos o palabras fuertemente relacionadas; por ejemplo, el concepto **artefacto** se halla relacionado o cuenta con los siguientes sinónimos (distribuible, documento, producto).

Un término o una frase puede tener más de una acepción en nuestro contexto, por lo que lo indicamos con números.

Definiciones

abstracción. el proceso de identificar un conjunto de características y propiedades comunes de una colección de *objetos* o *entes*.

actividad. 1. (tarea) 2. es un *subproceso* que no requiere mas descomposición.

agente. el *ente* ejecutor de un *proceso*. El agente puede ser tanto un ente humano como una herramienta o dispositivo computarizado.

ambiente de ingeniería de software centrado en procesos. (ambiente -o sistema- de soporte a procesos de software).

ambiente de soporte a procesos de software. (ambiente de ingeniería de software centrado en procesos) es un entorno de trabajo que ofrece asistencia a los usuarios en la ejecución de un *proyecto de software*.

anchor, ancla (punto de partida) origen de los *enlaces*. Puntos de partida para el proceso de navegación, esto es, cuando se selecciona el anchor, sucede la navegación a otro *objeto* o espacio de *información*.

arquitectura de proceso (meta-modelo de proceso) una estructura conceptual útil para describir y relacionar a las diferentes *perspectivas* de un *modelo de proceso*.

artefacto. (distribuible, documento, producto) es el producto creado, evolucionado, mantenido o destruido durante el *proceso de desarrollo de software* ya como un resultado requerido por un *agente* o para facilitar la prosecución de otro *proceso*. Es un *objeto* persistente que representa al producto de realizar una *tarea*.

atributo. 1. lo que se le atribuye de lo que es propio de un *objeto* o *ente*. 2. estado, variable, contenedor de *información* 3. *dato* miembro de una *clase*, estructura de dato 4. en *OOHDM* es un *dato* miembro de una *clase* del *modelo conceptual* y puede ser multitipado.

clase. (entidad, objeto, instancia) 1. representación abstracta de un conjunto de objetos que exhiben semejante comportamiento y propiedades. 2. La clase es el molde a partir del cual se crean objetos o instancias.

condición de un proceso es la declaración del estado de situación que debe ocurrir para el inicio, ejecución y finalización de un *proceso*. Una *actividad* puede comenzar cuando se cumple un conjunto de precondiciones y puede finalizar cuando se alcanzan las postcondiciones establecidas.

configuración de cambios. es una estrategia - asociada a algún *método*- para identificar, mantener, y administrar cambios de *artefactos* bajo cierta configuración. Se debe considerar aspectos de procedimientos de aceptación y congelamiento de los cambios, ítems configurables, versionamiento, entre otros asuntos.

constructor de proceso (método) es un *enfoque* específico o *método* que se puede usar para realizar una *tarea* o conjunto de tareas en dominios semejantes.

contexto de navegación. es una primitiva o *patrón de diseño* que está compuesta por un conjunto de *nodos*, *enlaces* y otros contextos (posiblemente anidados). Este *constructor de proceso* permite representar unidades cohesivas de conceptos y establecer relaciones semánticas apropiadas favoreciendo la orientación del usuario en la aplicación.

dato. (atributo) ítem elemental o primitivo de distinta naturaleza o medios, que sirve para contener y/o comunicar *información* elemental -con muy bajo nivel de elaboración.

descripción de proceso es un manera de representar y especificar la secuencia parcial de *actividades* de un *proceso*. Una descripción completa de proceso debe considerar las actividades y las *operaciones* asociadas, las *condiciones* para cada actividad, y a otros entes intervinientes en el *proceso de software* como *artefactos*, *agentes* y *roles*.

distribuible. (producto, artefacto, documento) es un *artefacto* solicitado por algún *proceso* o *agente* interno o externo.

documento. (producto, artefacto, distribuible) es un *artefacto* que puede contener o no información sobre otros artefactos.

elemento de proceso. (subproceso, paso de proceso)

enfoque. (estrategia)

enlace. 1 un vínculo entre *objetos* de navegación 2. En *OOHDM* los enlaces son la realización navegacional de las relaciones definidas en el *modelo conceptual*.

ente, entidad (clase, objeto) cualquier cosa, tangible o intangible, que está o exhibe comportamiento en el mundo real. El mundo real puede ser el mundo físico -hombre, computadora, teléfono-, o un mundo intangible o imaginario -sociedad, *proyecto de software*-.

estrategia. (enfoque) define las características más generales y representativas de un *proceso* de desarrollo, sus *principios* fundamentales y los *objetivos* y *metas* a alcanzar.

etapa. (fase, paso) *tareas* fundamentales de un *proceso de desarrollo*.

experticia. conjunto de habilidades específicas de un *agente*, surgido del estudio, del conocimiento y de la práctica.

evento. alguna ocurrencia que provoca el cambio de estado de un *objeto*.

fase. (etapa) es una agrupación de *procesos de software* fuertemente relacionados o cohesivos realizados en cierto orden. Las distintas fases pueden exhibir comportamientos diferentes.

GQM del inglés Goal-Question-Metric approach y se traduce en enfoque Meta-Pregunta-Métrica. Dado un conjunto seleccionado de *metas* del proyecto en el contexto de una organización -teniendo en cuenta las características y *atributos* deseables de los *artefactos*, *productos* y *recursos*, se construye y refina un conjunto de preguntas para cada meta y, en función de cada pregunta se eligen las *métricas* apropiadas.

heurística. *principio*, criterio o regla práctica, surgida de la *experticia*.

hipermedia. 1. es un conjunto organizado de *información* de diferentes medios, vinculados por relaciones estructurales y semánticas. 2. es la ciencia que se ocupa de estructurar, presentar y permitir acceso directo al contenido y relaciones, en un espacio organizado de *información*.

información. interpretación y elaboración de los *datos*, en cierto contexto

instancia. (objeto) una ocurrencia específica de una *clase*

lenguaje de modelado de procesos es un paradigma capaz de representar a un esquema de *meta-modelo de proceso* o alguna de sus *perspectivas.* Como todo lenguaje debe proveer una sintaxis precisa y una semántica no ambigua y amplia.

meta. (objetivo) representa a un conjunto de declaraciones de los resultados que se desean alcanzar en el contexto de la estrategia organizacional. Los resultados pueden estar en función de *artefactos*, *procesos*, etc.

meta-modelo de proceso. (arquitectura de proceso)

método. 1. (constructor de proceso) modo específico de realizar una *tarea*. Curso de acción u *operaciones* y conjunto de estándares y procedimientos de modelado a usar para tratar con alguna parte de un *proyecto de software.* 2. implementación de un servicio u *operación* de una *clase*

metodología. conjunto de *métodos* asociados a un *enfoque* con el fin de cubrir una o más *fases* o una parte significativa de una fase de un *proyecto -de software-*

métrica. es un valor numérico computado a partir de un conjunto de *datos* observables y consistentes con la intuición, y para que sea válido debe poseer un conjunto de características entre las que podemos enumerar: robustez, objetividad científica, permitir escalas y límites, relevancia, facilidad en la recolección de datos y validación.

modelo es una representación abstracta (abstracción) de *entes* o fenómenos de la realidad en la que se consideran los aspectos relevantes de los mismos y se desechan los menos relevantes sin que por ello deje de representar significativamente a esa realidad. Es una estructura en un dominio usado para representar a *entes* de otro dominio, con el propósito de comprenderlo y/o controlarlo.

modelo conceptual en *OOHDM* es una representación del dominio del problema construida con *clases*, relaciones, subsistemas y, eventualmente con *instancias*. Las clases pueden contener *atributos* multitipados.

modelo de proceso de producto. (modelo de proceso de software, modelo de ciclo de vida).

modelo de ciclo de vida. (modelo de proceso de software, modelo de proceso de producto).

modelo de proceso de software. (modelo de ciclo de vida) 1. es una *estrategia* apropiada para abstraer, organizar, ejecutar y/o controlar a las distintas *fases*, *tareas*, *recursos* y *artefactos* de un *proyecto de software* con el objeto de alcanzar las *metas* establecidas. 2. una descripción más o menos formal del *proceso de desarrollo de software*. Por lo que un modelo de proceso expresa: a) un cierto nivel de *abstracción*; b) una *perspectiva* particular del *proceso de desarrollo*.

modelo de proceso de hipermedia. 1. es un *modelo de proceso* para abstraer, organizar, ejecutar y/o controlar a las distintas *fases*, *tareas*, *recursos* y *artefactos* de un *proyecto de software de hipermedia* con el objeto de alcanzar las *metas* establecidas. 2. una descripción más o menos formal *del proceso de desarrollo de software* de *hipermedia*.

motor de procesos de un *ambiente de soporte a procesos de software* es el intérprete automático de las *descripciones de proceso* provistas de un modo estático o dinámico. Con dinámico queremos significar que durante la ejecución del proceso la descripción puede cambiar o evolucionar.

modelo de seguimiento. representa a una correspondencia consistente entre *artefactos*, permitiendo conocer de dónde se deriva un artefacto y cuáles otros se derivan de él.

nodo. 1. es una *clase* navegacional que contiene elementos de *información*, esencialmente *atributos* y *puntos de partida* o *anchors*. 2. En *OOHDM* un nodo representa una vista lógica de las clases definidas en el *modelo conceptual*.

objetivo. (meta) representa a un conjunto de declaraciones de los resultados que se desean alcanzar en el tiempo. Los resultados pueden estar en función de *artefactos*, *procesos*, etc. El objetivo, con respecto al tiempo, es de menor plazo que la meta y debe ser cuantificable.

objeto. (entidad, clase) 1. es la representación de un comportamiento que ocurre en el mundo real. Un objeto exhibe un comportamiento y se le atribuyen propiedades o *atributos* 2. una instancia u ocurrencia de una clase.

OOHDM que en inglés se traduce en Object-Oriented Hypermedia Design Method, es una *metodología* basada en *modelos* y *principios* de Orientación de Objetos útil para la especificación y construcción de *artefactos* de *hipermedia*.

operación. es una acción específica que se puede invocar para realizar una tarea. -En plural- conjunto de servicios que dispone una *clase* para responder a solicitudes externas.

paso, paso de un proceso. 1. (subproceso, elemento de proceso) 2. (etapa)

perspectiva. (vista, submodelo) es un enfoque particular para especificar y comunicar información del *modelo*.

principio. una regla general o verdad fundamental que puede servir de guía en la toma de decisiones.

proceso. (proceso de software) 1. *ente* usado para construir *artefactos*. 2. *ente* que mediante la función correspondiente, transforma entradas en salidas.

proceso de desarrollo de software, proceso de desarrollo es la realización de un *proyecto de software.*

proceso de software. (proceso) un conjunto parcialmente ordenado de *subprocesos* a los que se le asocian una colección de *recursos*, *agentes*, *condiciones*, *artefactos* y *constructores de proceso*, con el fin de producir los *distribuibles* conforme a las *metas* establecidas.

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 usuarios y desarrolladores. Un prototipo es una implementación parcial de todo el sistema o de componentes del mismo.

prototipación es una *estrategia* de desarrollo que ayuda a descubrir y especificar requerimientos, alternativas de diseño y la construcción de una base evolutiva del futuro sistema. Se caracterizada por un buen número de iteraciones y concurrencia con otras *actividades*, por un alto grado de participación de los usuarios, y un uso extensivo de *prototipos* y herramientas de producción avanzadas.

proyecto de software es un *ente* que comprende a un conjunto de *tareas*, tanto técnicas como de gerenciamiento, a un conjunto de *recursos*, a un conjunto de *estrategias*, *métodos* y *heurísticas*, con el propósito de lograr los *objetivos* y las *metas* acordadas.

proyecto de software de hipermedia es un *proyecto de software* con *recursos*, *tareas constructores de proceso* y *roles* específicos de la disciplina de *hipermedia.*

punto de partida (anchor, ancla)

recurso. es un *ente* necesario para que las *tareas* de un *proyecto de software* se puedan efectuar. Recursos de un proyecto son: humanos, monetarios, materiales, tecnológicos, temporales.

rol. es un conjunto de permisos y obligaciones que se debe asociar a un *agente* durante la realización de un tipo de *tarea*. El agente debe tener un conjunto de permisos para realizar las actividades de la tarea conforme a la submeta establecida y obligado a satisfacer un conjunto de *condiciones*.

RMM que en inglés se traduce en Relationship Managment Methodology, es una *metodología* basada en un *enfoque* estructurado y en el *modelo* de Entidad-Relación útil para la especificación y construcción de *artefactos* de *hipermedia*.

submodelo. parte o componente de un *modelo*

subproceso. (paso de proceso, elemento de proceso), cualquier *ente* resultante de la descomposición recursiva de un *proceso*. El nodo raíz del árbol es el *proceso* mas general y abstracto, el nodo hoja es el *proceso* más específico y lo denominamos *actividad*.

tarea. 1. (actividad) representa una unidad de trabajo a realizar por un *agente* 2. un *proceso* al que se le asocian componentes de gestión, es decir, se le pueden asignar *agentes*, *recursos*, se la puede planificar, programar, ejecutar y controlar. Las tareas manuales solo involucran a *agentes* humanos; las tareas automáticas solo dependen de *agentes* computarizados, en tanto que las tareas interactivas necesitan de ambos tipos de *agentes*.

técnica procedimientos y *heurísticas* específicas usadas por un *método*.

vista. (perspectiva, submodelo)

Referencias

[Basili et al. 84] **Basili, V.R., Weiss, D.**, 1984, "*A methodology for Collecting Valid Software Engineering Data*", IEEE Transactions on Software Engineering, Vol. 10 (3), pp. 728-738.

[Basili et al. 94] **Basili, V.R., Caldiera, C., Rombach, H.D.**, 1994, *"Goal Question Metric Paradigm"*, Encyclopedia of Software Engineering, Vol. 1, John Wiley & Sons, pp. 528-532 .

[Berard 93] **Berard, E.** ,1993, *"Essays on Object-Oriented Engineering",* Vol 1. Prentince Hall.

[Boehm 88] **Boehm, B.**, 1988, "*A Spiral model of Software Development and Enhancement",* IEEE Comp. 21, 5 (May88).

[Booch 94] **Booch, G.**,1994, *"Object-Oriented Analysis and Design with Application",* Benjamin/Cummings.

[Booch 96] **Booch, G.** ,1996, *"Object Solution: Managing the Object-Oriented Project",* Benjamin/Cummings

[Botafogo 92] **Botafogo, R. Rivlin, E., Shneiderman, B.**,1992, *"Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics".* ACM Transactions on Office Information Systems, 10(2), pp. 142-180.

[Collins 95] **Collins, D.**, 1995, "*Designing Object-Oriented User Interfaces",* Benjamin/Cummings.

[Conklin 87] **Conklin, J.**, 1987, "*Hypertext: an introduction and survey",* IEEE Comp. 20, 9, pp.17-40

[Connell et al 95] **Connell, J.L.; Shafer, L.**, 1995, "*Object-Oriented Rapid Prototyping*", Prentice Hall.

[Curtis et al 92] **Curtis, B.; Kellner, M.; Over, J.**, 1992, "*Process Modelling",* Comm. ACM 35, 9; pp. 75-90.

[Davis 92] **Davis A.**, 1992, "*Operational Prototyping: a new development approach",* IEEE Software 9 ,5 (Nov 92) pp.70-78

[Deming 86] **Deming, W.**, 1986, "*Out of Crisis*", MIT Center for Advanced Engineering Study, Mass.

[DoD 88] **US Department of Defense,** *Military Standard: Defense System Software Development. DOD-STD 2167A* Washington, D.C. Feb 1988.

[Dowson et al 91] **Dowson, M. ; Nejmeh, B.; Riddle, W.** 1991, "*Fundamental Software Process Concepts",* Proceed. First European Workshop on Software Process Modelling, AICA Press.

[Feiler et al 93] **Feiler, P. H**; **Humphrey, W.** 1993, "*Software Process Development and Enactment: Concepts and Definitions",* Proceed. Second Int. Conf. On the Software Process, US (Feb. 93).

[Fenton 91] **Fenton, N.E**., 1991, *"Software Metrics a Rigorous Approach",* Chapman & Hall, London.

[Fenton 96] **Fenton, N.E**., 1996, "*Do standards improve product quality?*", IEEE Software, 13(1), pp. 22-24.

[Garzotto et al 91] **Garzotto, F.; Paolini, P.**, **Schwabe, D.;** 1991, "*HDM, a model for a design of Hypertext Application"*, Proceed. of Hypertext'91, ACM Press.

[Garzotto et al 93] **Garzotto, F.; Schwabe, D.; Paolini, P.**, 1993, "*HDM, a model based approach to Hypermedia Application Design "*, ACM Transaction on Information System, Vol. 11, 1, Jan 93, pp. 1-26.

[Gilb 88] **Gilb, T**., 1988, *"Principles of Software Engineering"*, Adisson-Wesley.

[Goldberg et al 95] **Goldberg, A.; Rubin, K.**, 1995, "*Succeeding with Objects: decision frameworks for project management"*, Addison-Wesley.

[Grønbaek et al 94] **Grønbaek,K.; Trigg, R.H.**, 1994, "*Design issues for a Dexter-based hypermedia system",* Comm. ACM 37, 2 (Feb94) pp. 40-49

[Humphrey 88] **Humphrey, W.**, 1988, "*Characterizing the Software Development Process: A Maturity Framework",* IEEE Software, 5,2 (March88).

[Humphrey et al 89] **Humphrey, W.S., Kellner**, **M.I.** 1989, **"***Software Process Modelling: Principles of Entity Process Models"*, Proceed. of the 11th Int. Conference on Software Engineering, IEEE Comp. Society.

[Henderson et al 90] **Henderson-Sellers, B; Edwards, J.**, 1990, "*The Object-Oriented systems lifecycle",* Comm. ACM 33, 9.

[IEEE 93] **IEEE Recommended Practice for Software Requirements Specifications**, 830-1993 Standard

[Isakowitz et al 95] **Isakowitz,T.; Stohr, E.; Balasubramanian, P.**, 1995, "*RMM: a methodology for structured hypermedia design",* Comm. ACM 38, 8 (Aug 95) pp. 34-48

[Jacobson et al 92] **Jacobson, I.; Christerson, M.; Jonsson, P.; Overgaard, G.**, 1992, "*Object-Oriented Software Engineering: a use case driven approach"*, Addison-Wesley

[Jacobson 94] **Jacobson, I.**, 1994, "*Scenario-based Design",* J. Caroll Ed. ACM Press, Ch 12 : pp. 309-336.

[Lange 94] **Lange, D.**, 1994, "*An Object-Oriented design method for hypermedia information system",* Proceed. of the 27th Annual Hawaii International Conference on System Science.

[Lonchamps 93] **Lonchamps, J.**, 1993, "*A Structured Conceptual and Terminological Framework for Software Process Engineering ",* ICSP 2, Berlin, IEEE Press.

[Lyardet et al 96] **Lyardet, F. ; Rossi, G. ;**, 1996, "*Bridging the gap between hypermedia design and implementation. A research prototype",* Poster session during Hypertext96', Washington.

[Madhavji 91] **Madhavji, N.H..** 1991, "*The Process Cycle",* IEEE Software Engineering. Journal, (Aug 91).

[Nanard et al 91] **Nanard, J.; Nanard, M.**, 1991, "*Using Structured Types to Incorporate Knowledge in Hypertext",* Proceed. of Hypertext'91, ACM Press, pp. 329.

[Nanard et al 95] **Nanard, J.; Nanard, M.**, 1995, "*Hypertext Design Environment and the Hypertext Design Process",* Comm. ACM 38, 8 (Aug 95) pp. 49-56

[Olsina et al 95] **Olsina, L.; Nicolau, S.; Irastorza, J.; Bertone, E.**, 1995, "*Estrategias y Criterios de Diseño e Implementación Hipermediales en el proyecto Facultad de Ingeniería",* Informe de proyecto de I&D, Dep. de Informática, Fac. Ingeniería, UNLPam.

[Olsina 96] **Olsina, L.**, "*View of a Process Model to Develop Hypermedia" (in Spanish),* Proceed. of the IV Congress of the SCCC (Computer Science Chilean Society), Valdivia, Chile, 1996.

[Olsina 97a] **Olsina, L.**, 1997, "*Systematic use of Flexible Process Model to build Hypermedia Artifacts".* Poster Sesion, Hypertext 97, Southampton, England

[Olsina 97b] **Olsina, L.**, 1997, "*Object-Oriented Flexible Prototyping to support Hypermedia Flexible Process Model".* III Workshop em Sistemas Multimídia e Hipermídia (WoMH 97), pp. 3-14, Sao Carlos, Brasil.

[Olsina 97c] **Olsina, L.**, 1997, "*Applying the Flexible Process Model to build Hypermedia Products".* Hypertext and Hypermedia: Tools, Products, Methods (HHTPM 97), Paris, France.

[Olsina 98] **Olsina, L.**, 1998, "*Functional View of the Hypermedia Process Model"*. The Fifth International Workshop on Engineering Hypertext Functionality at ICSE´98, Kyoto, Japan.

[Osterweil 87] **Osterweil, L**. 1987, *"Software Processes are Software Too"*. Proceed. of the Ninth International Conference of Software Engineering, pp2-13, Monterey CA.

[Rossi et al 95] **Rossi, G. ; Schwabe, D.; Lucena C.J.P. ; Cowan, D.D.** , 1995, "*An Object-Oriented design Model for Designing the Human-Computer Interface of Hypermedia Application",* Proceed. of the International Workshop on Hypermedia Design (IWHD95), Springer Verlag .

[Rossi 96a] **Rossi, G.**, 1996, "*Uma metodologia Orientada a Objetos para o projeto de aplicativos Hipermídia",* Tesis Doctoral, PUC-RIO, Río de Janeiro, Br.

[Rossi 96b] **Rossi, G.**, 1996, "*Design Patterns for Object-Oriented Hypermedia applications. Patterns Languages of Program Design II",* Addison-Wesley **.**

[Rossi et al 97] **Rossi, G., Schwabe, D.; Garrido, A.**, 1997, "*Design reuse in Hypermedia Application Development",* Hypertext 97, Southampton, England pp 57-66

[Royce 70] **Royce W**., 1970, "*Managing the Development of Large Software System"*, IEEE WESCON pp 1-9. Reprinted in Nineth IEEE International Conference on Software Engineering, Washington DC: Computer Society Press of IEEE, 1987 pp.328-338.

[Rudd et al 94] **Rudd, J; Isensee, S.**, 1994, "*Twenty-two tips for a happier, healthier prototype",* Interaction Vol 1,1 (Jan 94) pp. 35-40

[Rumbaugh et al 91] **Rumbaugh, J; Blaha, M; Premerlani, W; Eddy, F; Lorensen, W. ,**1991, "*Object-Oriented Modelling and Design*", Prentice Hall.

[Schwabe et al 94] **Schwabe, D.; Barbosa, S.** , 1994, "*Navigational Modelling in Hypermedia Application",* Technical Report MCC 42/94, Dep. de Informática PUC-Rio, Brasil

[Schwabe et al 95a] **Schwabe, D.; Rossi, G.** , 1995, "*Building Hypermedia Application as Navigational Views of Information Model",* Proceed. of the 28th Hawaii International Conference on System Science, Jan 95, Vol 3, pp. 231-240

[Schwabe et al 95b] **Schwabe, D.; Rossi, G.** , 1995, "*The Object-Oriented Hypermedia Design Model",* Comm. ACM 38,8 (Aug 95) pp. 45-46

[Schwabe et al 96] **Schwabe, D.; Rossi, G. Barbosa, S** , 1996, "*Systematic Hypermedia Application Design with OOHDM",* Hypertext 96, US

[Thüring et al 95] **Thüring, M.; Hannemann, J.; Haake, J.** , 1995, "*Hypermedia and Cognition: Designing for Comprehension",* Comm. ACM 38, 8 (Aug 95) pp. 57-66

[Wirfs-Brock et al 90] **Wirfs-Brock, R.; Wilkerson, B; Wiener, L.**, 1990, "*Designing Object-Oriented Software*", Prentice Hall