OOWS: Un Mètode Dirigit per Models per al Desenvolupament d'Aplicacions Web

TESI DOCTORAL

Joan Fons i Cors

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Directors:

Dr. Vicent Pelechano Ferragud
Dr. Òscar Pastor López

Departament de Sistemes Informàtics i Computació

Març de 2008

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

OOWS: Un Mètode Dirigit per Models per al Desenvolupament d'Aplicacions Web

TESIS DOCTORAL

Presentada per:

Joan Fons i Cors

Dirigida per:

Dr. Vicent Pelechano Ferragud
Dr. Òscar Pastor López

València, Març de 2008

A Mirèia, la meua vida ...

"**El plaer que acompanya al treball**
fica en oblit la fatiga."
Horacio.

Agraïments

Aquesta tesi no és sols meua, és de tots aquells que han cregut en mí des del primer moment. També és d'aquells que no han perdut la paciència, veient passar els mesos al calendari.

Us agraïsc a tots vosaltres que m'heu ajudat a llevar les pedres del camí per que el meu pas fos més ferm i lleuger. A tots aquells que heu vist aquesta tesi acabada quan jo encara tenia els ulls tancats. A tots els que m'heu dibuixat un somriure oferint-me esperança. Als que heu encés un llum a la meua obscuritat. A aquells que començàreu amb mí, que encara seguiu ahí i que sé que seguireu tota la vida. A aquells que m'han esperat sense importar el temps perquè sabien que a pesar del retràs, arribaria. I ací estic ....

Als que mai m'han abandonat i sobretot a aquells que em buscaven i no em trobaren ... sols dir-vos, gràcies!

En primer lloc vull donar les gràcies el Prof. Vicent Pelechano, *Pele*, per mostrar-me el respecte, la sinceritat i la humiltat amb que s'ha d'afrontar el món de la investigació. Per ser un apassionat investigador, un professional excel·lent i una gran persona. Per haver estat sempre al meu costat i per la teua confiança cega. Per a mí eres més que el director d'aquesta tesi, eres també el meu germà gran i un amic dels de debò.

Al Prof. Òscar Pastor, per haver-me donat la meua primera oportunitat de començar en aquest món de la investigació, i per transmetre'm

l'esperit d'evangelitzar el món a colp d'OO-Method. Però sobretot, per la seua vessant humana i per ser més un colega que un director.

A Pedro, Vicky, Manoli, Gonzalo i Javi, per ser els companys de fatigues durant tots aquestos anys, i per les nombroses discussions trascendentals tractant d'ajudar-nos mútuament.

A Paco, per haver-me ajudat a dur a la pràctica aquestes idees.

A Carlos, Pau i Fani, per la seua incessant ànsia d'ajudar.

A Magalí i Isabel, que encara que estan lluny, sempre han estat ben aprop.

A la resta del grup OO-Method (Juan, Jorge, Ana, Marta, Sergio, Nelly, Paqui, Sònia, Ignacio, Juan Luis, Bea, Giovanni i Luis) per ser la millor gent amb que hom podria trobar-se per treballar dia a dia.

Als meus pares, Juan José i Encarna, perquè al mirar-los sempre he vist en ells dedicació, confiança, comprensió, recolzament i sobretot amor. Ells són en gran mesura causants que haja arribat fins ací.

Als meus germans Vanesa i Jordi, perquè amb ells visc moments de segona infància on s'obliden les tensions dels adults.

A Vicent, que amb respecte i simpatia s'ha fet un espai en la família.

A la meua altra família, Ascensión i Eva, per haver-me acceptat com part de la seua.

A Paqui i Pablo, per tants bons moments, per la vostra amistat i estima.

A Mirèia. Per ser la meua companya en aquest llarg camí, i la meua inspiració. Perquè amb un somriure i un gest aconsegueixes donar-me pau i tranquilitat.

Moltes gràcies!

RESUM

Internet s'ha convertit per mèrits propis en el mitjà de comunicació per excel·lència. La rapidesa en la que es pot intercanviar la informació unida a l'eliminació de les barreres geogràfiques i tecnològiques han convertit Internet en la plataforma preferida per divulgar el coneixement i fer negocis.

Per tal de dur a terme el desenvolupament d'aplicacions de programari en aquestos entorns, han sorgit nombroses aproximacions que segueixen el que ve anomenant-se com Enginyeria Web. Aquestes aproximacions defineixen processos de desenvolupament i extensions conceptuals basades en models, orientat al desenvolupament de programari web.

D'altra banda estan els principis del Desenvolupament Dirigit per Models, on es proporcionen marcs conceptuals i entorns que permeten construir processos de desenvolupament amb generació de codi a partir dels models.

Sota aquesta perspectiva s'enmarca aquesta tesi, en la que es defineix OOWS, un entorn de producció de programari per al web que aplica els principis proposats per l'Enginyeria Web i el Desenvolupament Dirigit per Models. Aquest entorn defineix un procés de desenvolupament que empra uns models conceptuals extesos amb característiques web i una estratègia per obtenir automàticament l'aplicació web a partir d'aquestos models OOWS.

Per donar suport a aquesta aproximació, s'ha desenvolupat una eina

que permet editar i gestionar aquestos models conceptuals web i que a més a més implementa les regles de transformació que permeten obtenir un prototipus de l'aplicació web a partir d'aquestos models.

RESUMEN

Internet se ha convertido por méritos propios en el medio de comunicación por excelencia. La velocidad con la que se puede intercambiar información unida a la eliminación de las barreras geográficas y tecnológicas han convertido a Internet en la plataforma preferida para divulgar el conocimiento y hacer negocios.

Para llevar a cabo el desarrollo de aplicaciones software en estos entornos, han aparecido numerosas aproximaciones que aplican lo que se conoce como Ingeniería Web. Estas aproximaciones definen processos de desarrollo y extensiones conceptuales basadas en modelos, orientadas a la construcción de aplicaciones web.

Por otro lado están los principios del Desarrollo Dirigido por Modelos donde se proporcionan marcos conceptuales y entornos que permiten construir processos de desarrollo con generación de código a partir de modelos.

Bajo esta perspectiva se enmarca esta tesis, en la que se define OOWS, un entorno de producción de software para la web que aplica los principios que se proponen en la Ingeniería Web y en el Desarrollo Dirigido por Modelos. Este entorno define un proceso de desarrollo en base a unos modelos conceptuales extendidos con características web y una estrategia para obtener automáticamente la aplicación web a partir de estos modelos OOWS.

Para dar suporte a esta aproximación se ha construido una herra-

mienta que permite editar y gestionar estos modelos conceptuales web, y que implementa las transformaciones que permiten obtener un prototipo de la aplicación web a partir de estos modelos.

ABSTRACT

Internet has become the main communication platform. Then way in which the information can be massively reached without geographical or technological issues has leaded this platform to be the preferred mechanism to share knowledge and to do business.

Many approaches have appeared to develop software applications for web environments, following the Web Engineering princicples. These approaches are mainly focused on defining appropriate web development processes by using conceptual model extensions to describe web artifacts.

In the other hand, Model Driven Software Development has emerged to provide conceptual frameworks and environments to build development processes using code generation capabilities.

Taking these influences as basis, this thesis presents OOWS, an environment for web software development that applies the princples of the Web Engineering community with the Model Driven Software Development principles. This environment defines a development process that uses web extended conceptual models, and defines a strategy to build a web application taking as input OOWS web conceptual models.

It has been developed a tool to support this approach. This tool allows to edit and manage OOWS conceptual models, and implements the code transformation rules to obtain a prototype of a web application.

Índex

Índex de figures

Índex de taules

Índex de Codis Font

Capítol 1

Introducció

El creixement d'Internet, les Intranet, les Extranet i el World Wide Web (la Web), ha causat un impacte significatiu en tots els sectors de la vida social així com en la nostra vida personal i en els nostres estils de treball. La Web obri possibilitats que mai abans en la recent història de la informàtica s'havien tingut i davant aquestes alternatives ens trobem que molts dels sistemes d'informació existents, així com les bases de dades legades, s'estan migrant cap a la Web i Internet.

Internet s'ha convertit per mèrits propis en el mitjà de comunicació per excel·lència. La rapidesa en la que es pot intercanviar la informació unida a l'eliminació de les barreres geogràfiques, han convertit a Internet en la plataforma preferida per les empreses per extendre els seus negocis. A causa d'açò estan proliferant el número d'aplicacions en la Web per a la resolució de les necessitats de les organitzacions.

A diferència de les aplicacions tradicionals basats en llenguatges compilats i interfícies gràfiques, les aplicacions web estan basades en estàndars amplament acceptats (HTTP i HTML), que poden ser emprants pràcticament en qualsevol tipus de dispositiu. Cada vegada més, els desenvolupaments Web i les tecnologies associades a aquestes, s'estan emprant com una plataforma d'implementació per a diferents àmbits

(institucions, educació, aplicacions gubernamentals, financeres, etc...). En tots aquestos casos, a més de la complexitat de la funcionalitat que han d'oferir, s'uneix la complexitat del desenvolupament en entorns distribuits. Ací, Internet altra vegada, proposa una solució distribuida que facilita aquestos desenvolupaments.

Es podria suposar que els nous reptes i paradigmes que implica la Web estan també implicant nous mètodes per suportar els processos de creació i migració de sistemes de dades a la mateixa. Tanmateix, això no és així. El ben cert és que l'enfoc de desenvolupament actual de les aplicacions i sistemes basats en la Web segueix sent "*ad-hoc*" i "creatives", sense processos disciplinats, sistemàtics i rigurosos que permeten garantitzar l'obtenció de productes de programari de qualitat. Al revés, els productes adoleixen de la necesitat d'una gran quantitat de correccions (també "*ad-hoc*") que donen orige a sistemes que, encara que avuí en dia continuen operant, no existeix la garantia d'açò mateix en el futur, pel fet de no estar dissenyats correctament amb termes com ara mantenibilitat o portabilitat propis de la qualitat de programari.

La naturalesa de la Web ha evolucionat des d'aquella el propòsit de la qual era únicament distribuir informació, fins l'actual, que exigeix gran quantitat de funcionalidad ofertada pels sistemes tradicionals i que ara s'han dut a la Web. Açò ha conduït al sorjiment de l'Enginyeria Web com l'estratègia encaminada a abordar les característiques de desenvolupament que aquestos sistemes ofereixen, d'una forma metodològica, repetible i correcta. La tendència actual dels sistemes basats en la Web es cap a la complexitat i sofisticació, de tal manera que existeix una legítima preocupació al voltant de la manera en la qual estan siguent desenvolupats, així com en la seua qualitat i integritat. Davant l'ausència de processos disciplinats de desenvolupament plenament assentats, en un futur no molt llunyà ens haurem d'enfrontar amb els problemes de desenvolupament, intalació, operació i manteniment d'aquestos sistemes

"*ad-hoc*" .

En resum: els sistemes basats en la Web que en gran nombre s'estan desenvolupant en l'actualitat, davant un escalament (per exemple), tenen una alta probabilitat de fallida. I el que és encara més preocupant, és que dia a dia, la nostra vida està depenent cada vegada més d'aquest tipus de sistemes, de tal manera que fallides que puguen presentar-se en els mateixos tindran greus repercusions que lamentar. Aquest problema fou batejat temps enrere com "La crisi de la Web" [80]

Com fer front a aquest crisi de la Web? [81] ens indicà que és indispensable un enfoc disciplinat així com nous mètodes i eines per al desenvolupament, intalació i avaluació dels sistemes basats en la Web. Aquestos enfocs deuen considerar els següents aspectes dels sistemes basats en la Web: 1) les seues característiques úniques; 2) el seu ambient operacional; 3) L'aparició de múltiples escenaris, i 4) la diversitat de perfils d'usuari que involucra la construcció de sistemes basats en la Web. Tot açò anterior implica nous reptes en el desenvolupament d'aplicacions basades en la Web.

Motivats per totes aquestes premises, molts experts han unit els seus esforços en la creació i establiment d'una nova disciplina, anomenada *Enginyeria Web*. Aquesta s'enfoca exclusivament en l'estudi i la solució dels problemes que sorgeixen a partir dels sistemes basats en la Web. [81] la defineix com: *L'establiment i ús de principis d'enginyeria i gestió, així com d'enfocs sistemàtics i disciplinats per al seu desenvolupament, intalació i manteniment exitós de sistemes i aplicacions d'alta qualitat basats en la Web*. [98], ho resumeix així: *En un lloc Web existeix més que un disseny visual i interfície d'usuari. Els locs Web estan consolidant-se més com programes i menys com documents estàtics*. Queda clar que la idea de la qual sorgí la Web canvià d'èsser mèrament un mecanisme divulgador d'informació cap a convertir-se en vertaders sistemes d'aplicació amb funcionalitat, abans no concebida per a la Web.

En resum, l'Enginyeria Web és la *branca de l'Enginyeria del Programari que estudia els processos, mètodes, tècniques i recursos esencials per al desenvolupament d'aplicacions Web de qualitat.*

Tot model de processos d'Ingenieria Web deuria poseir un conjunt de primitives conceptuals que li permeten representar els diferents aspectes que les seues aplicacions requereixen. No existeix en l'actualitat un acord definitiu al respecte de l'ontologia que deuria conceptualitzar qualsevol especificació d'una aplicació Web.

S'observa que en els models de processos d'Enginyeria Web per a l'elaboració de sistemes per a aquestos entorns web, existeiexen diferents aproximacions o corrents principals: aquelles que parteixen d'un disseny de base de dades i extenen les seues funcionalitats amb característiques hipermedials i de comportament, aquelles que inicien un anàlisi hipermedial (navegacional) del sistema i el completen amb informació d'estructura d'informació i comportament i finalment aquelles que agafen com a punt de partida un anàlisi Orientat a Objectes (OO) i extenen aquest amb característiques hipermedials.

1.1 Problemàtica i Motivació

L'utilització d'aplicacions Web per a resoldre el problema de les organitzacions és ventajós en nombrosos sentits. De cara a l'usuari, sols li cal un *navegador web* i els coneixements bàsics per emprar-lo. No cal que assimile els coneixements d'instalació i/o configuració d'artefactes de programari específic, i el seu ús serà similar al que podria proporcionar una aplicació via Web. A causa de l'adopció d'estàndars acceptats, se soluciona el problema tecnològic imposat per les plataformes destí (sistema operatiu, entorn, etc.), podent ser emprada la mateixa aplicació web en diferents entorns, sense haver d'adaptar-se. Inclús, amb l'arribada de dispositius mòbils amb més capacitats (com ara els telèfons mòbils d'última

generació, les PDAs o ordinadors ultra-compactes), que inclouen també navegadors web, aquestes aplicacions arriben també fàcilment a aquest tipus de dispositius. D'altra banda, el manteniment d'una aplicació es simplifica, fet que només canviant la versió del servidor (controlada per l'entitat proveedora de continguts), tots els clients, sense realitzar cap canvi, tenen accés immediat al nou sistema.

Actualment s'està investigant com poder emprar noves metodologies de desenvolupament per generar aplicacions Web de manera sistemàtica. Però, la gran majoria d'aplicacions web desenvolupades han sigut desenvolupades mitjançant mètodes tradicionals o tecnologies propietàries. Aquestes aplicacions sofreixen greus problemes de manteniment, qualitat i reusabilitat. A causa que una aplicació Web no és més que un producte de programari, és indispensable emprar les pràctiques de l'*Enginyeria del Programari* per solucionar aquesta problemàtica. Per desgràcia, les metodologies tradicionals no estan adaptades als requeriments específics de les aplicacions Web.

L'*Enginyeria Web* sorgeix amb l'objectiu d'aplicar els fonaments de l'Enginyeria de Programari sobre els sistemes Web, i ha sigut ben acceptada en la comunitat científica, on han aparegut nombrosos treballs, foros de discussió especialitzats, conferències sobre al respecte i xarxes temàtiques.

Es pot considerar que hi ha un acord en l'estratègia bàsica de treball. Així, si repassem a grans els processos de producció, les etapes involucrades, els models conceptuals per a la web, les estratègies d'implementació, etc., no hi trobem grans diferències.

Malgrat, cada aproximació ha aportat la seua perspectiva dins d'aquest entorn. A l'entrar a major nivell de detall és quan comença a notar-se la diferència: algunes aproximacions donen major importància a certes fases o etapes del procés, els models conceptuals de base emprats per extendre cap a models conceptuals web, la definició o no

d'unes transformacions dels models cap al codi, la sistematització o no d'aquestes transformacions, són alguns dels exemples que hi aparèixen.

Especial menció s'ha de fer al tema de la navegació. Pareix clar, i totes aquestes aproximacions ho accepten, que una etapa primordial dins la conceptualització dels sistemes web és la de la definició d'un model d'hipermèdia. Aquest model, més conegut com **model navegacional**, és l'encarregat de definir l'accessibilitat al sistema per als usuaris, especificant els "camins de navegació" vàlids en el sistema. Malgrat açò, encara *no hi ha consens en la definició del concepte de navegació ni en la ontologia emprada per representar-la.*

Altre aspecte molt rellevant és el grau de sistematització en l'obtenció de la solució final (codi). Es pretén aconseguir una **generació automàtica de codi a partir dels models concpetuals web** descrits, seguint les aproximacions de prototipació automàtica. És per això que serà necessari emprar *models formals de referència* i definir una sèrie de regles de transformacions entre els elements de modelatge conceptuals en l'espai del problema a components de programari en l'espai de la solució.

El *Desenvolupament de Programari Dirigit per Models* (DSDM) comença a proporcionar resultats prometedors. Existeixen diferents indicadors que ens fan ser optimistes en quan a l'evolució e implantanció industrial d'aquesta filosofia de desenvolupament. Començant per la gran empempta que constitueix la proposta MDA (*"Model Driven Architecture"*) [86] de la OMG, i la recent aparició de les Fàbriques de Software [61]. Potser un dels aspectes més esperançadors és el desenvolupament continu de tecnologies i eines per la construcció d'eines CASE que donen suport a processos DSDM, como són el projecte *"Eclipse Modeling Project"* [41], les *"Domain-Specific Language"* (DSL) Tools [78] i altres eines que també empren aquest paradigma per al desenvolupament d'aplicacions, com ara el *ArcStyler* [60], Together [16] o AndroMDA [9].

El món de l'Enginyeria Web també s'apunta a aquesta tendència i di-

verses aproximacions ha anat apareguent amb la finalitat de proporcionar un suport per al desenvolupament d'aplicacions Web dirigit per models. Aquestes aproximacions introdueixen models conceptuals per capturar de forma abstracta els diferents aspectes que defineixen una aplicació Web. A partir d'aquestos models, proposen l'ús de transformacions model-a-model i/o model-a-text amb la finalitat d'obtenir el codi equivalent a la representació abstracta de l'aplicació Web que constitueix el model. En aquest sentit, cal destacar aproximacions com OOHDM [118], WebML [28], WSDM [36], UWE [54], OOH [74], Hera [55] o OOWS [44], l'aproximació que es presenta en aquest treball de tesis docotoral, que són amplament acceptades en l'àmbit científic de l'Enginyeria Web i que han sigut provades i validades en diferents casos d'estudi. A més a més, la majoria d'aquestes aproximacions ja proporcionen eines de suport al mètode que proposen, com és el cas de WebRatio (WebML), ArgoUWE (UWE) o HyperDe (OOHDM). Aquestes eines permete la definició de models que capturen els aspectes estructurals, navegacionals, de presentació i d'adaptativitat de les aplicacions web. Algunes inclús, faciliten el desenvolupament del codi de l'aplicació. Tanmateix, encara que estan en constant desenvolupament, els aspectes de comportament són actualment suportats de manera parcial, des d'un punt de vista de modelatge conceptual.

Aleshores, en aquesta tesi s'aborda el problema de definir un entorn de desenvolupament de programari que done una solució concreta per a la construcció d'aplicacions Web seguint el paradigma que proposa l'*Enginyeria Web*, aplicant els principis del *Desenvolupament Dirigit per Models* i les *Factories de Programari*, implementant una eina de suport que facilite aquesta tasca.

1.2 Contexte de la Tesis Doctoral

Aquest treball tesi ha sigut desenvolupat dins el grup d'investigació "OO-Mehtod: Mètodes de Producció de Programari OO", que pertany al Departament de Sistemes Informàtics i Computació de la Universitat Politècnica de València (DSIC). Una de les principals tasques en aquestos últims anys ha sigut la d'extendre el propi mètode OO-Method per donar-li capacitats per a diferents aplicacions i entorns tecnològics.

El treball presentat en aquesta tesis forma part d'uns dels treballs realitzats a OO-Method en el que l'autor ha participat activament en la seua concepció i desenvolupament científic. A causa de l'èxit dels resultats obtesos per aquesta tesis, han sorgit nombroses extensions a aquest treball per tractar de donar-li un alcanç encara major. A nivell de resultats comercials, aquest treball ha servit com un dels pilars per la definició de *OlivaNOVA*, un producte de desenvolupament industrial d'aplicacions de programari, creat per CARE Technologies S.A. i que dóna una implementació concreta al mètode OO-Method.

Els treballs que han possibilitat el desenvolupament d'aquesta tesis s'enmarquen en els següents projectes d'investigació i desenvolupament:

- "Ingeniería de Requerimientos y Generación Automática de Software". DGEUI (Direcció General d'Ensenyaments Universitaris i Investigació, Conselleria de Cultura, Educació i Ciència) amb referència: GV97-TI-05-34 (des de 1998 fins 2000)

- "Generación Automática de Sistemas Software en Ambientes Orientados a Objetos". Projecte CICYT del programa FEDER, amb referència: TIC 1FD97-1102 (des de Octubre de 1999 fins Desembre 2001)

- "WEST: Web Oriented Software Technology". Projecte CYTED VII-18 (Programa Iberoamericà de Ciència i Tecnologia per al

Desenvolupament (des de Agost del 2000 fins Agost 2003)

- "WEE-NET: Web Engineering Network of Excellence". European Comission (Alph European Union Project) (des de 2004 fins 2007)
- "TAISSI: Tecnologia Software Avanzada para la Ingenieria del Software de la Sociedad de la Información", "Subproyecto Ingeniería de Ambientes Web", Projecte MCYT amb referència: TIC 2001-3530-C02-01
- "DESTINO: Desarrollo de e-Servicios para la nueva Sociedad Digital", Projecte MCYT amb referència: TIN 2004-03534

1.3 Objectius

El principal objectiu d'aquesta tesi doctoral és definir *un procés de desenvolupament d'aplicacions web complet i sistemàtic*, seguint els principis marcats per l'Enginyeria Web.

Aquest procés de desenvolupament ha d'estar basat en la *definició de models conceptuals adequats i adaptats a sistemes web*, de manera que siguen aquestos models els que guien i dirigisquen la producció de les aplicacions web.

Dit d'una altra manera, convertir aquestos models conceptuals web en la peça clau i principal de tot el desenvolupament, extenent els models conceptuals çlàssics"de manera adient. Els models, segons les estratègies del Desenvolupament Dirigit per Models passen a ser els elements principals del software.

Degut a la naturalesa de constant evolució i adaptació a requeriments canviants dels sistemes web i la necessitat de cicles de desenvolupament àgils, sistemàtics i rigorosos, s'ha optat per desenvolupar el treball dins l'àmbit dels paradigmes de prototipació automàtica.

A més a més, es pretèn construir una extensió d'una aproximació Orientada a Objectes que capture d'una manera natural aspectes d'estructura estàtica i de comportament. Aquest model Orientat a Objectes s'ha consolidat com una aproximació més adequada per a representar una realitat amb dinàmica que no pas el model Entitat Relació.

Allò que es pretén aconseguir amb la tesis doctoral és extendre el un mètode de producció d'aplicacions de programari per capacitar-lo i adaptar-lo a la producció d'aplicacions de programari per al web, emprant com a marc de treball les pautes de l'Enginyeria Web. Per a fer-ho, s'emprarà l'aproximació OO-Method com a base, ja que aquesta aproximació aporta tots els requeriments necessaris: (1) empra models conceptuals on és té en compte d'una manera adequada tant l'estructura estàtica com dinàmica del sistema; (2) proporciona un entorn de generació de codi que aplica els principis del Desenvolupament Dirigit per Models; i (3) poseeix una eina industrial, *OlivaNOVA*, que implementa de manera completa el mètode OO-Method.

Per tant, els objectius a aconseguir són:

- Afinar el procés de producció de programari "clàssic" adequant-lo i adaptant-lo per produir aplicacions web
- Aconseguir, amb ajuda d'OO-Method, una solució completa de producció d'aplicacions web dins un paradigma de prototipació automàtica
- Introduir un model de navegació i un model de presentació per captar les propietats web i relacionar-los amb els ja existents en OO-Method (Diagrama de Classes, Model Dinàmic i Model Funcional)
- Definir el conjunt de primitives d'aquests nous models tenint en compte aspectes com senzillesa, nivell d'abstracció i intuitivitat

- Estudiar a fons les relacions entre models per intentar aprofitar al màxim el coneixement que es pot deduir inter-models i que puguen guiar en l'especificació de les propietats "web"
- Introduir una etapa per a fer una gestió explícita dels diferents tipus d'usuaris del sistema
- Introducció de mecanismes de reutilització a nivell d'especificacions conceptuals navegacionals
- Definir el conjunt de patrons de traducció que hauran d'aplicar-se per transformar automàticament els models conceptuals web en prototipus d'aplicacions web
- Construir un prototipus d'una eina CASE que facilite i guie el procés de desenvolupament d'aquests sistemes web i que implemente els patrons de traducció a entorns d'implemetació
- Aconseguir una integració funcional *OlivaNOVA*, l'eina que implementa els principis proposats per OO-Method
- Aplicar, des de l'inici, els principis promoguts per l'Enginyeria Web, el Desenvolupament Dirigit per Models i les Factories de Programari

Finalment, s'ha tingut en compte, durant el pas del temps, l'aparició de noves disciplines o línies d'actuació que han pogut influenciar l'evolució de l'estratègia, com ara:

- L'aparició de MDA com estàndar de documentació de transformacions de codi en aproximacions basades en el MDD
- La renaixença de la necessitat de construir models dominis ontològics específics, molt relacionats amb temes de comerç elctrònic, suportat en gran mesura per la Web Semàntica

- La necessitat actual de donar suport a models de processos complexos i oberts, i l'evolució de la tecnologia cap a entorns de serveis web i arquitectures orientades a serveis
- El fort impacte que la comunitat d'hipermedia adaptativa està tenint en l'àmbit de l'adaptativitat de sistemes basats en el web

1.4 Solució Proposada

La proposta que es presenta s'ha anomenat *OOWS*, un mètode de producció d'aplicacions i Solucions Web basat en una aproximació de modelatge OO. OOWS proposa una extensió d'OO-Method que introdueix un refinament al procés de producció per adeqüar-lo a les necessitats de modelatge conceptual i desenvolupament d'aplicacions web.

La Figura 1.1 presenta la proposta d'extensió del procés de desenvolupament d'OO-Method (remarcat amb un 1) per extendre'l (remarcat amb un 2) adequadament.

Les extensions que es proposen són:

1. La incorporació d'un **Model Navegacional** que permet definir l'estructura de navegació per als usuaris del sistema. En l'àmbit d'aquest model navegacional es proposen dues etapes: una primera en la que es detecten i es representen adequadament els diferents tipus d'usuaris que poden emprar el sistema (mitjançant un diagrama d'usuaris) i una segona en la que es descriga l'accés al sistema per a cadascun d'aquest tipus d'usuari. Aquest model navegacional ha de ser construït agafant com a entrada el model estructural d'OO-Method per tal de crear les diferents vistes enllaçades del sistema que se li proporcionarà als diferents usuaris.
2. La incorporació també d'un **Model de Presentació**, que capture a alt nivell d'abstracció els requeriments de presentació d'informa-

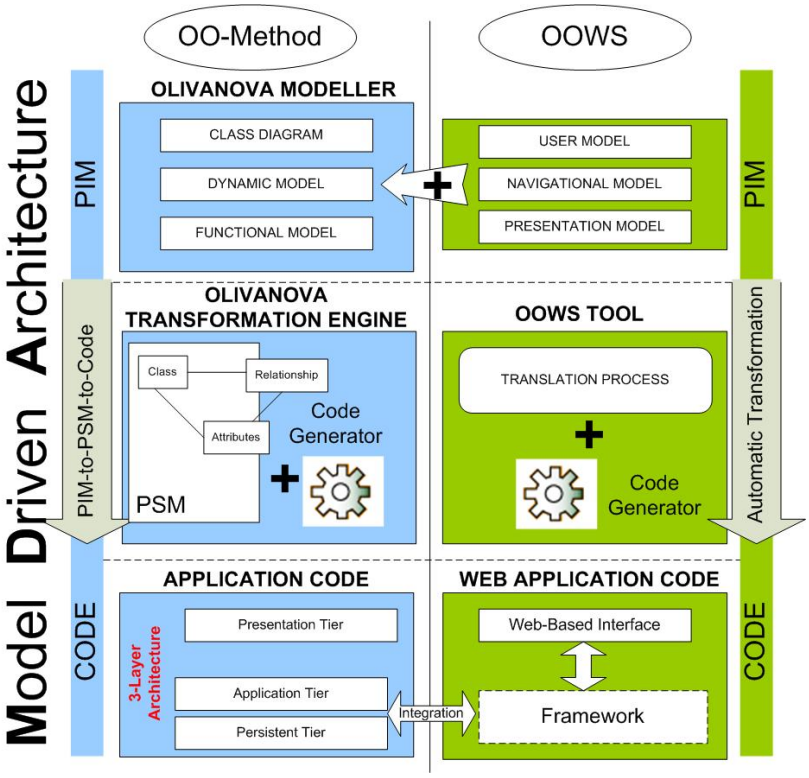


Figura 1.1 – Aproximació Metodològica OO-Method/OOWS

ció orientat a aplicacions web. Es proposa que aquest model siga construït agafant com a base el model navegacional descrit anteriorment, amb l'objectiu de "decorar" les primitives navegacionals amb informació sobre cóm presentar-les.

3. Extendre el **procés de generació de codi** d'OO-Method i incorporar nous patrons de traducció per a aquestes extensions conceptuals per al web. A nivell d'arquitectura, canviar la capa d'Interfície Gràfica d'Usuari que s'obté amb OO-Method per una altra dedicada i construïda seguint els requeriments del web.

4. Definir un **entorn de generació de codi per aplicacions Web**, basat en un Framework d'implementació. Aquest framework ha sigut el resultat d'aplicar els principis del Desenvolupament Dirigit per Models i les Factories de Programari. Aquest framework està pensat per que puga integrar-se amb les solucions que s'obtenen amb *OlivaNOVA*, la implementació comercial d'OO-Method.

5. Desenvolupar un prototipus d'**eina** que done **suport complet** al desenvolupament d'aplicacions Web amb el **mètode OOWS**, integrant-se amb l'eina *OlivaNOVA* existent.

L'objectiu final serà, com s'ha comentat abans, aconseguir un mètode de producció d'aplicacions web a partir de models conceptuals web que fan ús de l'orientació a objectes en entorns de producció automàtica de software.

1.5 Estructura de la Tesis Doctoral

Aquesta tesi doctoral està dividida en els següents capítols:

- El *Capítol 1 - Introducció* presenta el context de la tesi doctoral, introduint la problemàtica en el desenvolupament d'aplicacions web en l'àmbit de l'Enginyeria Web i els objectius essencials a resoldre.

- El *Capítol 2 - Sistemes Software per al Web* defineix els artefactes de programari per a la web i presenta l'estat de l'art de les aproximacions metodològiques més recents que tracten de donar solució al desenvolupament d'aquest tipus de sistemes.

- El *Capítol 3 - OOWS: Un Mètode per Desenvolupar Aplicacions Web* presenta el procés de desenvolupament complet d'una aplicació web, fusionant l'aproximació metodològica definida per OO-

Method amb les extensions conceptuals proposades per OOWS. Una primera fase on es descriu el sistema en termes del Domini del Problema (Anàlisi Conceptual) i una segona on es descriu el procés d'obtenir una implementació del sistema de manera automàtica en termes del Domini de la Solució (Desenvolupament de la Solució).

- El *Capítol 4 - Model Navegacional* descriu amb detall el model introduït per OOWS per capturar la semàntica navegacional de les aplicacions web. Es repasa l'etapa d'autoria a gruix nivell ("*authoring-in-the-large*") i la d'autoria al detall ("*authoring-in-the-small*") junt amb les primitives de modelatge que permeten aquesta representació.

- El *Capítol 5 - Model d'Usuaris* descriu els diagrames d'usuaris que permet la introducció i gestió explícita del tipus d'usuaris al sistema web. Es defineix la taxonomia d'usuaris, el seu mode d'accés i les relacions entre usuaris.

- El *Capítol 6 - Conceptes Avançats* presenta uns refinaments del Model de Navegació proposat per tal de donar suport a diferents aspectes avançats, com ara l'ús de mecanismes d'accés, la gestió avançada de continguts, la introducció d'aspectes de personalització en aplicacions web o la definició de mecanismes per a la reutilització d'especificacions.

- El *Capítol 7 - Model de Presentació* presenta una sèrie de patrons orientats a enriquir la descripció dels contextos navegacionals per expressar requeriments de presentació.

- El *Capítol 8 - Implementació de la Interfície Web* defineix el procés d'obtenció de la interfície web a partir de les extensions introduïdes per OOWS: Model d'Usuaris, Model Navegacional i Model de Presentació. Ací es presenten els patrons de traducció i

un framework d'implementació que s'ha desenvolupat per facilitar el desenvolupament de les aplicacions.

- El *Capítol 9 - OOWS Suite: Eina de Suport* descriu el prototipus d'eina de suport que s'ha desenvolupat i que dóna suport al procés de desenvolupament descrit en la tesi. Bàsicament presenta: (1) els editors gràfics construïts per representar els models conceptuals proposats i (2) els compiladors de models que implementen els patrons de traducció definits.

- El *Capítol 10 - Conclusions* presenta les conclusions d'aquest treball de tesi, analitza el grau amb que s'han aconseguit els objectius marcats i avalua el grau d'èxit amb respecte a publicacions, participacions en foros i xarxes temàtiques. Finalment, es presenten els treballs en curs i futurs.

- El *Annexe A - Framework d'Interfícies Web, WIF* descriu un framework d'implementació d'interfícies web que s'ha desenvolupat per facilitar la implementació de les aplicacions Web. Aquest framework permet simplificar el procés de compilació de models, ja que proporciona primitives d'implementació de major nivell d'abstracció. A més, és l'encarregat d'integrar-se amb la implementacions *OlivaNOVA*. Aquest apèndix detalla la interfície d'ús d'aquest framework.

- El *Annexe B - Metamodel* mostra el metamodel d'OOWS emprat tant per definir amb precisió les primitives de modelatge, com per donar suport a la construcció de l'editor de model i la base per definir les transformacions de models.

- El *Annexe C - Cas d'Estudi: IMDb Lite* mostra el cas d'estudi d'IMDb, desenvolupat de manera completa amb OOWS. IMDb és una aplicació web per a la gestió filmogràfica, proporcionant

informació sobre pel·lícules, actors, directors, etc. del món del cine.

- El *Annexe D - Un Compilador de Models d'OOWS* mostra el codi d'un compilador que agafa models conceptuals OOWS i els transforma en codi del framework d'interfícies Web. Aquest compilador està implementat emprant oAW, una tecnologia de construcció de compiladors de models que permet realitzar transformacions de model a codi.

Capítol 2

Sistemes Software per al Web

2.1 Introducció

Uns anys després de l'aparició de la disciplina l'Enginyeria Web [81], el desenvolupament d'aplicacions web complexes és encara un tòpic de recerca relevant. Durant molts anys, les aproximacions enginyerils web han realitzat un excel·lent treball adaptant mètodes de desenvolupament de programari [99] que inicialment foren concebuts per donar suport al desenvolupament de programari "tradicional" (no web) per tal de proporcionar una solució concreta al desenvolupament d'aplicacions web.

Tots aquestos mètodes d'Enginyeria Web estan basats en un principi similar: les aplicacions web deuen construir-se a partir de descripcions precises, robustes i no ambigües del Sistema d'Informació en forma d'Esquemes Concpetuals. Després, aquestos Esquemes Conceptuals deuen transformar-se adientment en la seua representació en forma de producte de programari, definint lligams entre les primitives conceptuals i les seues representacions en tecnologies concretes. Per tal d'aconseguir aquest objectiu, els Esquemes Conceptuals clàssics, que bàsicament es centren en capturar els aspectes estàtics i de comportament del sistemes, han de ser extesos amb nous models i mecanismes d'abstracció per tal de

capturar els nous aspectes introduïts per les aplicacions web. Aquestos aspectes "nous" poden resumir-se bàsicament en dos: navegació i presentació. Alguns esforços representatius per introduir-los en tècniques de modelatge conceptual tradicionals han sigut OOHDM [105, 119], WebML [29], UWE[65, 64] and WSDM [37, 35].

En aquest contexte, algunes aproximacions recents d'Enginyeria Web han fet avanços significatius en seguir estratègies Dirigides per Models [76, 11, 122]. Aquest fet es fa més evident si considerem que alguns mètodes d'Enginyeria Web han adoptat de manera exitosa la proposta **Model-Driven Architecture** (MDA) proposada per la OMG [86]. D'acord a aquesta proposta MDA, els Esquemes Conceptuals són comparats amb Models Independents de Plataforma (PIM). Després, aquestos PIMs són transformats en Models Específics de Plataforma (PSM) per l'aplicació de transformacions model-a-model.

La proposta que es proposa en aquesta tesi doctoral proporciona una contribució concreta en aquest contexte. S'ha adaptat el mètode OOWS que segueix la disciplina de l'Enginyeria Web per que satisfaga els principis de MDA [137]. OOWS proposa un PIM que ens permet descriure els diferents aspectes de les aplicacions web. Aquest PIM exten l'Esquema Conceptual del mètode Orientat a Objectes de producció de programari OO-Method [94, 93, 95] nous models per capturar la "semàntica web".

El PIM proposat pel mètode proporciona tota la informació necessària per dur a terme les transformacions automàtiques. Aquestes transformacions es realitzen mitjançant una eina que transforma el PIM en codi executable. Açò pot ser realitzat a causa que el PIM és computacionalment complet . Per realitzar aquestes transformacions, s'ha extés el procés de generació de codi definit per OO-Method [90] incloent aquelles referents a les extensions introduïdes.

2.2 La Web: Present i Futur

La Web és atractiva. No sols per a experts en computació, sinó també per als usuaris no experts. Açò és demostra en la creixent quantitat d'usuaris que empren els serveis de la Web. Al mateix temps, s'observa una millora en la relació usuari-computadores gràcies a la Web: els camps de les ciències de la computació, els sistemes d'informació i l'enginyeria de programari, que abans quedaven ubicades en l'àmbit dels experts, amb la Web han alcançat l'usuari comú. I no perquè resulte trivial abordar-los, sinó més bé per la disponibilitat d'eines que permeten la construcció "relativament fàcil" dels llocs Web. Açò ha conduït cada vegada més a l'usuari cap al desenvolupament artesanal de llocs Web [38].

Però, quines implicacions té aquest accés tan fàcil a eines de desenvolupament Web? La resposta és sencilla: podem trobar cada vegada més i més usuaris amb perfils, formacions i orientacions diferents a la branca de la computació, que estan contribuint a enriquir-la, com ara experts en disseny gràfic, en gestió documental i d'hipertext, en màrketing, en aspectes legals, etc. Tot açò fa, com s'expresa en [98]: *Els sistemes basats en la Web estan involucrant una barreja entre edició publicitària i desenvolupament de programari, entre màrketing i computació, entre comunicacions internes i relacions externes, entre art i tecnologia*.

Altre fenòmen interessant que s'ha presentat és aquell que té a veure amb la perspectiva de l'usuari que té el desenvolupador. Tal i com ho es pretén amb aproximacions cap al Desenvolupament Dirigit per l'Usuari ("*End-User Development*"[70]), un dels aspectes que deu considerar tot dissenyador de l'interfície de qualsevol sistema ha de ser: *Conèixer a l'usuari*. Tanmateix, l'*usuari* en la Web és un concepte potser massa ampli: podem ubicar-lo dins una organització, en la Intranet; podem ubicar-lo fora de l'organització, en la Extranet; o enlloc, en Internet. Tots amb perfils, interessos i característiques diferents que s'afigen un

problema addicional de personalització de llocs Web que augmenta la seua complexitat [38].

En definitva, la complexitat és tal que es requereixen diferents habilitats que no es solen trobar en un mateix expert: Enginyeria del Programari, d'Hipertext, d'Informació, de Requeriments, d'Anàlisi i Disseny de Sistemes, de Modelatge i Simulació, d'Administració de Projectes, de Proves, d'Interacció Home-Màquina, de Multimedia, ... entre d'altres. És per açò que es requereixen d'aproximacions metodològiques que aporten una solució enginyeril (sistemàtica i reproduible) per tal de facilitar aquest tipus de desenvolupaments a la vegada que es millora la qualitat del producte produït.

A mesura que augmenta la nostra habilitat en la construcció de sistemes Web, augmenta també la seua complexitat. El número de professionals que treballaran directament en assumptes Web del futur serà cada vegada major i en conseqüència, la complixitat de la seua coordinació també ho serà. Conforme els desenvolupadors vagen adoptant els principis que promou l'Enginyeria Web, es reduiran també els riscos causats per aquesta complexitat i es millorarà la taxa d'èxit en els desenvolupaments.

Com la mateixa Web, l'Enginyeria Web enfronta i enfrontarà la complexitat del canvi constant. L'Enginyeria Web deurà ser capaç d'adaptar-se a aquesta dinàmica també. És natural: si la Web canvia de manera accelerada, l'Enginyeria Web també ho haruà de fer. Sols així podrà fer front de manera real i exitosa als reptes i problemes, avuí desconeguts, que oferirà aquesta nova àrea tecnològica.

2.3 Característiques inherents als Sistemes Web

Els sistemes Web actuals[1] són híbrids entre hipermedia i sistemes d'informació. Com als sistemes hipermedials, la informació és accedida d'una forma exploratòria, on la manera en què la informació és navegada i presentada és de gran importància. Pel que fa a la seua perspectiva de sistema d'informació, el tamany i la volatilitat de les dades i la distribució de les aplicacions requereix de solucions d'arquitectures consolidades basades en tecnologies com sistemes de gestió de bases de dades i computació client-servidor [46].

A causa d'aquesta naturalesa híbrida, el desenvolupament d'una aplicació Web deu tenir en compte un conjunt de requeriments d'aplicació, com ara:

- La necessitat de gestionar tant dades estructurades (per exemple, registres de les bases de dades) com informació no estructurada (per exemple, objectes multimedia).
- Donar suport a un accés exploratori de les dades a través d'interfícies navegacionals.
- La personalització de l'estructura de continguts, primitives navegacionals i estils de presentació.
- El suport a un comportament proactic, per exemple, per fer recomanacions i filtrats de dades.

Pel que fa al seu desenvolupament, encara que els Sistemes Web i els Sistemes d'Informació "tradicionals" involucren ambdós conceptes de programació i desenvolupament de programari, la construcció de sistemes

[1] alguns autors solen anomenar-los tambe "*Web Information Systems, WIS*" o símplement aplicació Web

basats en el Web és diferent al desenvolupament de software tradicional amb relació als següents aspectes:

1. La majoria dels sistemes Web actuals són orientats al document, incloent documents (pàgines Web) estàtics i dinàmics.

2. Existeix un èmfasi molt marcat en la creativitat dels aspectes visuals de la presentació i la interfície.

3. Molts sistemes basats en el Web segueixen sent dirigits pel contingut.

4. Molts sistemes basats en el Web deuen gestionar la diversitat de perfils d'usuari que poden emprar-los. Si en alguna cosa caracteritza un sistema Web és en la universalitat d'usuaris potencials que poden emprar-lo.

5. Encara no s'ha explotat al màxim, malgrat tot, la capacitat del Web com el medi natural d'obtenir aplicacions funcionals distribuides, més que com un medi de distribuir continguts. En la actualitat s'està fent un esforç en proporcionar serveis ofertats via web accessibles de manera estàndar.

6. Existeix una combinació d'art i ciència en la Web de manera que mai abans s'havia considerat en els sistemes tradicionals de programari.

7. La dinàmica del Web és tan elevada que és necessari que els temps de canvis i posada en funcionament han de ser cada vegada més reduïts.

8. El Web opera en una diversitat de plataformes, que no és comú amb el software tradicional de plataforma única.

9. Els individus que participen en la construcció de sistemes basats en la Web requereixen d'habilitats diverses i variades que influeixen de manera decisiva en la forma en que es concebeix el Web, així com en els seus aspectes de qualitat.

Com ja ha esdevingut en el passat amb altres tecnologies emergents com ara les bases de dades i els llenguatges de programació orientats a objectes, les metodologies i les eines software poden proporcionar una ajuda molt significant en la gestió adequada de la complexitat d'aquest tipus d'aplicacions, donant un suport a l'enteniment i ús d'aquest paradigma de desenvolupament i proporcionant avantatges en la productivitat i reducció de riscos inherents en el desenvolupament de programari i migració de sistemes [46].

2.4 L'Enginyeria Web

Motivats per aquest context, molts experts estan unint els seus esforços en la creació i establiment d'una nova disciplina, anomenada Enginyeria Web, que s'enfoca exclusivament a l'estudi i la solució dels problemes que sorgeixen a partir dels sistemes basats en el Web. [81] la defineix com: *L'establiment i ús de principis d'enginyeria i gestió, així com d'enfocs sistemàtics i disciplinats per al desenvolupament, instalació i manteniment exitós de sistemes i aplicacions d'alta qualitat basats en el Web.*

El primer pas per a sistematitzar el desenvolupament d'aplicacions web complexes, amb aquestes necessitats d'evolució, extensibilitat i manteniment, és definir un procés de *cicle de vida* adequat. Algunes característiques desitjables d'aquest procés és que siga altament iteratiu i incremental, amb l'objectiu de disminuir el temps de desenvolupament i millorar la qualitat del producte final.

De les estratègies que han anat sorgint per donar aquest suport

metodològic, s'observa que en els processos de construcció de sistemes Web existeixen dues corrents clarament diferenciades:

1. Una corrent en la que un sistema per al web és bàsicament un conjunt de documents d'hipertext enllaçats que conformen l'espai d'hipermedia. Però, sense incloure aspectes de "dinàmica" funcional. De fet, tendeixen a pensar que els sistemes web són tan complexos que no es pot realitzar un model conceptual complet que capture les seues propietats. Habitualment, aquestes aproximacions venen del món de la comunitat de la *Interacció-Home-Màquina* i l'*Hipermedia Adaptativa*. Aquestes estan centrades en el fet que els sistemes Web estan basats en models de navegació molt sofisticats i que inclouen d'una manera principal als usuaris i l'adaptativitat dinàmica dels sistemes a aquests usuaris.

2. Una altra corrent en la que es considera als sistemes Web com una evolució de les aplicacions de programari tradicionals on han agafat major importància cert tipus de característiques (disseny de documents-hipertext, navegació, gestió d'usuaris, introducció d'aspectes de presentació, etc.).

Pel que fa a la primera corrent, des del punt de vista del treball a realitzar en aquesta tesi queda descartada com estratègia base de treball, a causa que no es tracta de manera adequada totes les dimensions necessàries per tal de construir models conceptuals web (en concret la dimensió funcional i una mica de l'estàtica). D'altra banda està descompensat l'esforç que es fa en l'àmbit de definició de les propietats navegacionals. Malgrat tot, alguns aspectes d'aquestes aproximacions s'han emprat en la construcció de la solució metodològica aportada en l'àmbit de definició de propietats navegacionals i gestió d'usuaris.

Però, és la segona corrent la que ha desembocat en el naixement de l'Enginyeria Web [81] com l'estratègia per abordar els reptes que

aquests sistemes ofereixen, d'una manera metodològica, estructurada, sistemàtica i rigorosa. Es basa en el fet de pensar que sí que és possible construir un **model conceptual** que represente les característiques rellevants de les aplicacions **web**. I, com a conseqüència, que aquestos models conceptuals deuen constituir la base per guiar el desenvolupament de les aplicacions web.

Les aproximacions més importants que han aparegut al llarg d'aquests últims anys dins de l'Enginyeria Web (revisades al següent capítol) hi estan d'acord pel que fa a les característiques essencials:

1. Definició de processos rigorosos de conceptualització d'aplicacions web, semblants en les seues etapes.

2. Ús/Definició de models conceptuals extesos dels "tradicionals" per incorporar també els requeriments d'aquest tipus d'aplicacions.

3. Definició de guies metodològiques a l'hora d'implementar les solucions de programari a partir d'aquestos models conceptuals.

2.5 Aproximacions Metodològiques

A continuació es revisen les aproximacions que es consideren en l'actualitat més rellevants en l'àmbit de l'Enginyeria Web. A pesar que en són moltes i hi ha que venen d'altres branques de treballs, només s'analitzen aquells que apliquen els principis de l'Enginyeria Web i el Desenvolupament Dirigit per Models, ja que aquestes premisses són essencials en l'àmbit que aquesta tesi ocupa.

D'aquestes aproximacions s'analitza el procés de desenvolupament que proposen, així com els models que introdueixen per capturar les especificacions de les aplicacions Web. Observant les primitives i ordre en que es capturen, podrem veure quins conceptes són els que tracten de donar prioritat.

2.5.1 HDM

El Model de Disseny d'Hipertext de [50] i [49] és el primer model per al disseny estructurat de sistemes web. Ofereix una extensió del model entitat-relació afegint-hi un model de disseny hipermedial, però sense contemplar aspectes funcionals. S'ha decidit ficar-lo perquè fou el que inicià el "camí a seguir".

En realitat, aquesta aproximació no defineix el procés de desenvolupament d'una aplicació web, però, comenten els autors en [49] que pot ser integrat amb algún procés existent.

Un avantatge que ofereix és la seua possibilitat de descomposar les aplicacions d'hipertext en artefactes de gra més fi reutilitzables mentre s'asseguren les relacions estructurals amb el model entitat-relació. El principal inconvenient són les limitacions conceptuals que imposa el mètode.

És més un model de disseny que no pas un model conceptual. A més, l'analista està subjecte a una forta càrrega cognitiva acusada per l'absència d'un model de procés insatisfactori i incomplet que no suporta el reús d'artefactes i el fet de treballar amb un paradigma més clàssic com és l'entitat relació. Tampoc és trivial fer el "*mapping*" del disseny construït de les aplicacions d'hipertext a una aplicació web, a causa d'una manca de suport adequat.

La seua principal bondad és el fet de ser un dels primers mètodes en aparèixer i de servir d'inspiració per a la seua versió objectual OOHDM.

2.5.2 OOHDM

Emprant com a base HDM, naix el *Mètode de disseny hipermedial Orientat a Objectes OOHDM* [108, 117]. OOHDM ofereix un procediment per al desenvolupament d'aplicacions hipermedials consistent en cinc etapes: *El·licitació de Requeriments*, *Disseny Conceptual*, *Disseny Navegacional*, *Disseny Abstracte d'Interfície* i *Implementació*, els quals han de

construir-se seguint un enfoc iteratiu incremental, basat en prototipus.

En la fase d'**El·licitació de Requeriments**, en primer lloc s'identifiquen els actors ("*stakeholders*") i les tasques que aquestos han de dur a terme. A continuació es defineixen un conjunt d'escenaris per a cada tasca i tipus d'actor en forma de Casos d'Ús. Aquestos casos d'ús són descrits emprant una notació gràfica introduida pel mètode anomenada *Diagrames d'Interacció d'Usuaris* (UID) [140]. Aquestos diagrames especifiquen el fluix d'informació que ha de seguir i conduir a l'usuari. Aquestos UIDs són validats amb els actors i a partir d'aquest s'haurà de constuir el model conceptual bàsic.

En la fase de **Disseny Conceptual** s'ha de construir un model conceptual del domini de l'aplicació. OOHDM no força a l'ús de cap metodologia en particular, però es suggereix recolçar-se amb alguna Orientada a Objectes i coneguda, com pot ser OMT o UML [120].

En la fase de **Disseny Navegacional**, es suggereix emprar una metodologia basada en escenaris per guiar el procés, emprant una notació específica i incloent una seqüència d'etapes prou detallada, que passa per: 1) determinar el perfils o tipus d'usuaris i identificar les seues tasques; 2) recolectar els escenaris d'ús; 3) analitzar i produir una representació diagramàtica simple de les rutes de navegació definides com a conseqüència dels escenaris d'ús. 4) síntetitzar un diagrama de contexte parcial, especificant la navegació dins els contextes que suporten la tasca descrita en l'escenari. I finalment, 5) sintetitzar el diagrama de contexte final, a través d'un procés d'unió i homogenització dels esquemes parcials produïts en l'anàlisi d'escenaris del pas anterior. L'artefacte en el que s'expresa tot açò és el Model Navegacional. Aquest constitueix un descriptor de vistes navegacionals sobre el Model Conceptual.

En la fase de **Disseny Abstracte de l'Interfície** d'usuari no es proposa una guia metodològica concreta, però es donen una sèrie de decisions que l'analista pot anar tenint en consideració. OOHDM dis-

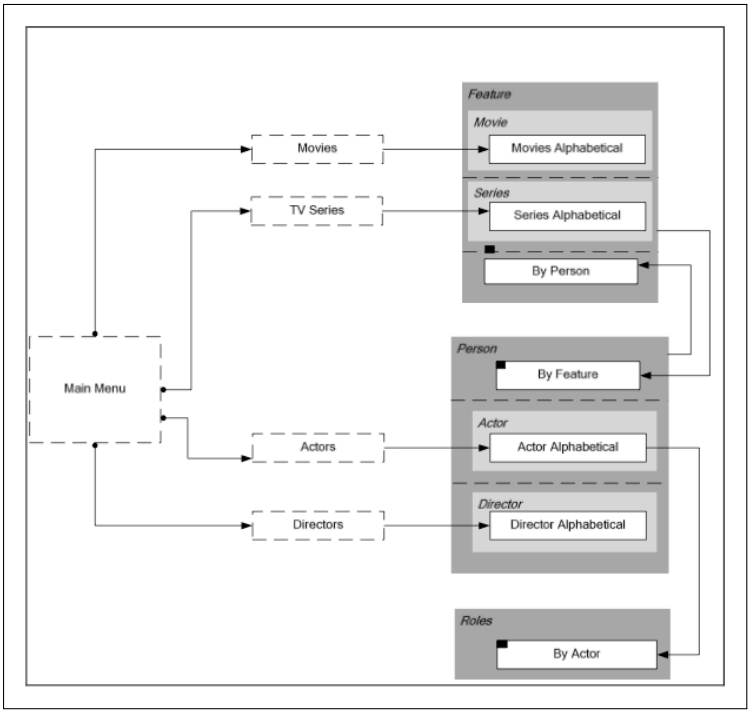


Figura 2.1 – OOHDM: Exemple Esquema Navegacional

tingeix dos nivells d'interfície: 1) *abstracta*, que defineix la interfície per a complir una tasca sense tenir en compte aspectes tecnològics, i 2) *concreta*, en el que es defineix el "*look and feel*" de l'aplicació (colors, fonts, etc.) i els elements gràfics que suporten l'interacció. La interfície abstracta es basa en un conjunt d'elements abstractes que permeten especificar una reacció a un event extern ("*SimpleActivator*"), mostrar contingut ("*ElementExhibitor*"), introduir una variable ("*VariableCapturer*") o composar-se entre sí ("*CompositeInterfaceElement*"). Cadascun d'aquestos elements deu ser associat a un element del model navegacional que proporcionarà o rebrà el contingut, i a un component concret d'interfície, que donarà una solució concreta en la plataforma tecnològica

destí triada. Aquesta associació es realitza mitjançant una especificació definida en OWL [123].

Actualment, la versió d'OOHDM ha sigut modificada cap a una altra versió, SHDM [121], en la que aquesta aproximació s'introdueix fortament en l'ús d'aspectes semàntics. Així, el Model d'Objects ara es deriva a partir d'una especificació d'un model de dades en RDF [20], que ha sigut proposat per descriure dades i meta-dades en l'àmbit de la Web Semàntica.

Finalment, la fase d'**Implementació** ofereix línies guia que permeten fer el "*mapping*" dels models anteriors a un ambient d'execució particular i que és independent d'una plataforma específica. En l'actualitat aquest mètode disposa d'una eina de suport al mètode, HyperDE, que s'analitza a la Secció 2.6.1.

Com a virtuts d'aquest mètode està el fet d'incorporar guies metodològiques que orienten l'analista en construir les especificacions, incloent en les últimes revisions activitats que permeten introduir certs aspectes d'enginyeria requeriments per al web. A més, el suport a la descripció de propietats navegacionals és molt exhaustiu i disposa de diferents primitives conceptuals per al tractament de casos especials com ara l'incorporació d'instàncies d'objectes a nivell de modelatge o el tractament de "perspectives" d'atributs.

Com a punts més febles destaca el fet que, si bé es defineix un procés guiat (mitjançant "*mappings*" entre elements de modelatge i implementacions) de producció de programari, no existeix una sistematiztació d'obtenció de codi. Açò ha sigut millorat amb el desenvolupament de l'eina HyperDE, que permet realitzar les transformacions de manera automàtica.

A més, encara que s'empra un paradigma Orientat a Objectes, no està clar amb quines tècniques es descriuen els processos funcionals i de vegades aquests pareixen confondre's amb aspectes de navegacionals.

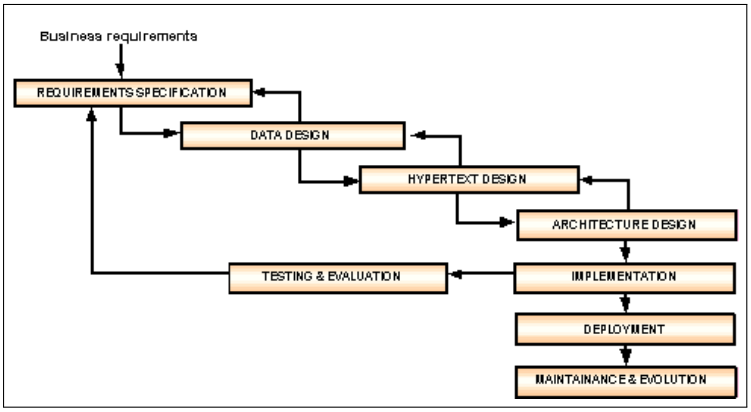


Figura 2.2 – WebML: Procés de Desenvolupament

2.5.3 WebML

En [28, 30] es proposa el Llenguatge per al Modelatge Web *WebML* [145] com "l'opció que permet als dissenyadors web expressar les característiques centrals d'un lloc Web sense preocupar-se pels seus detalls arquitectònics". Els conceptes inclosos en WebML s'associen a una representació gràfica intuitiva, factible de ser suportada per eines CASE i que poden ser emprats en fases de desenvolupament de codi.

L'especificació en WebML d'un lloc Web consisteix en quatre perspectives ortogonals: en primer lloc el *Model Estructural o de Dades*, que expresa el contingut de dades del sistema en termes de les entitats rellevants i les seues relacions (mitjançant un model entitat-relació extés). En segon lloc, el *Model d'Hipertext*, que descriu les vistes d'hipertext que poden publicar-se del lloc web. Aquestes vistes contenen submodels que permeten (1) composar les pàgines, (2) descriure la navegació entre pàgines i (3) donar pautes independents de tecnologia de la presentació de les pàgines. El *Model de Presentació* defineix l'aparença gràfica i la ubicació dels elements en la interfície. Finalment, el *Model de Personalització* que especifica els aspectes relatius als usuaris i els seus grups.

El **Model de Estructural o de Dades** expresa el contingut de l'aplicació Web en termes d'entitats i relacions. Les entitats actuen com contenidors dels elements d'informació mentre que les relacions estableixen conexions semàntiques entre les entitats. Cada entitat té atributs que representen les seues propietats amb el seu corresponent tipus de dada i poden organitzar-se (les entitats) de manera jeràrquica emprant l'operació de generalització[2]. D'altra banda, les relacions es defineixen mitjançant restriccions de cardinalitat i noms de rol entre les dues entitats.

La definició de la interfície del sistema s'especifica mitjançant el **Model d'Hipertext**, que es converteix en el model principal. Aquest model defineix l'estructura d'una aplicació web de manera modular, mitjançant vistes del sistema ("*site views*"), àrees, pàgines ("*pages*") i unitats de contingut. Una vista representa la interfície web (hipertext) que cobreix un subconjunt específic dels requeriments de l'aplicació. Una vista es composa d'àrees que formen les seccions principals de l'aplicació Web. Un àrea, de la seua banda, pot contenir pàgines o altres àrees (de manera recursiva), per tal de facilitar l'organització navegacional del sistema.

Es poden definir diferents vistes sobre el mateix model de dades, per tal de cobrir les necessitats d'un conjunt particular d'usuaris o per organitzar el contingut atenent a criteris diferents, com ara tecnològics. Un exemple clar és la de poder definir una vista específica pensant en un dispositiu concret, com podria ser una PDA, i crear un model que descriga aquesta aplicació.

Les pàgines són contenidors d'informació que es presenta a l'usuari. Aquestes s'organitzen mitjançant unitats de contingut ("*content units*"), que són peces elementals d'informació. Una unitat de contingut representa vàries instàncies d'entitat expresades en el Model de Dades. Aquestes

[2]El Model Entitat-Relacio subjacent que s'empra permet la definició d'estructures jeràrquiques de generalització entre les entitats.

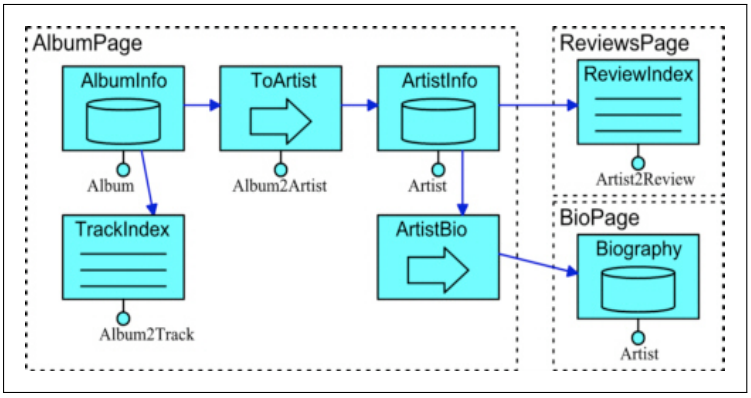


Figura 2.3 – WebML: Exemple Estructura Navegacional

unitats s'expresen mitjançant consultes o bé sobre atributs o relacions de les entitats. WebML proposa l'ús de diferents tipus d'entitats en funció de la seua utilitat: 1) Unitats de Dades, que mostren informació d'una única instància; 2) Unitats de Dades Múltiples, que mostren informació de múltiples instàncies; 3) Unitats d'Índex, que presenten una llista d'enllaços seleccionables; 4) Unitats d'"*Scroll*", que permeten navegar de manera ordenada sobre un conjunt d'instàncies; i 5) Unitats d'Entrada, que mostren un formulari per que l'usuari introduisca dades al sistema (generalment per executar una operació).

En WebML, tant les unitats de contingut com les pàgines poden interconectar-se mitjançant enllaços ("*links*"). Els enllaços entre unitats s'anomenen enllaços contextuals, ja que envien informació contextual (paràmetres de l'enllaç) de l'orige al destí. Els enllaços entre pàgines s'anomenen enllaços no contextuals, ja que no duen informació contextual.

La inclusió de serveis i operacions es realitza mitjançant operacions per defecte (crear, modificar i eliminar instàncies, afegir o eliminar relacions) associats a les entitats. També proporciona operacions essencials, com ara *login o logout*. És possible establir enllaços associats a l'execució

d'una operació, de manera que es puga navegar a un lloc o un altre en funció de si l'operació ha anat correctament (*ok*) o no (*ko*).

El **Model de Presentació** expresa l'aparença gràfica i la distribució dels elements de contingut dins les pàgines Web. Es poden distingir dos possibles tipus d'especificacions de presentació: a) específiques de pàgina, que inclouen referències explícites al contingut, o b) genèriques, que es basen en elements de contingut genèrics en totes les pàgines.

El **Model de Personalització** modela de manera explícita el concepte d'usuari i grups d'usuaris mitjançant entitats definides dins el Model Estructural. Els atributs d'aquestes entitats són emprats per definir expresions en OQL (Object Query Language) per establir polítiques de personalització, que després podrà ser reutilitzat en altres models.

A banda, WebML defineix un model de processos que involucra actors de diverses habilitats, com ara l'expert de dades, encarregat del model estructural, l'arquitecte de l'aplicació, que dissenya les pàgines i les rutes de navegació, l'arquitecte d'estils, que dissenya l'estil de la presentació i l'administrador que s'encarrega de la gestió dels usuaris i opcions de personalització.

Existeix una implementació d'una eina CASE, *WebRatio* [146] que suporta WebML. Aquesta eina agafa especificacions WebML i obté com a eixida la implementació tecnològica del lloc Web.

Al llarg dels anys, s'ha anat introduint expressivitat per abarcar altres àrees relacionades, com ara: el consum i publicació de serveis Web [18, 72], la definició de workflows per a l'execució de processos [17, 19] i la creació d'aplicacions Web sensibles al contexte i adaptatives [27].

Malgrat tot, aquesta aproximació té dues característiques essencials que han de ser millorats. La primera és el baix nivell d'abstracció al que es treballa. Moltes de les primitives que empra estan més orientades al disseny que no pas a l'especificació. Açò provoca que l'analista dels sistemes estiga continuament pensant en les "pàgines i enllaços" i que les

descripcions cresquen en complexitat.

Un altre inconvenient és la manca d'un model dinàmic (de comportament) clar. En WebML es poden descriure operacions com instruccions de consulta i modificació de la base de dades, que s'especifiquen mitjançant sentències SQL en les descripcions navegacionals.

2.5.4 WSDM

WSDM ("*Web Semantics Design Method*") [37, 36] és un mètode de disseny Web semàntic, bassat en la filosofia del disseny dirigit per l'audiència. Açò significa que els requeriments dels *usuaris objectiu* (audiència) són el punt de partida del procés de modelatge, i no els requeriments d'estructura d'informació o de comportament dinàmic, com sol ser més habitual.

El procés de desenvolupament (Figura 2.4 a la pàgina següent) consta de cinc fases: especificació de la *Missió de l'Aplicació*, *Modelatge de l'Audiència*, *Disseny Conceptual*, *Disseny de la Implementació* i *Implementació*.

En la fase de **Missió de l'Aplicació** es construeix una especificació de la "missió" o objectiu ("*Mission Statement*") del sistema Web. En aquesta descripció, en llenguatge natural, deuen ser formulats de manera precisa: 1) El propòsit general que persegueix el siste Web, 2) els distints tòpics que deuen ser coberts pel sistema, i 3) identificar els diferents tipus d'usuaris. Aquesta fase serà emprada com referència per a la resta de fases.

Una vegada establerta la missió del sistema, el següent pas consisteix en **Modelar l'Audiència** ("*Audience Modelling*"), on per a cada usuari del sistema es refina la missió del sistema adequant-la a aquest usuari. En primer lloc, tots els membres que tenen en comú requeriments d'informació i funcionalitat, són classificats en una classe d'audiència. Per tal d'identificar les classes d'audiència, es defineixen dos mètodes:

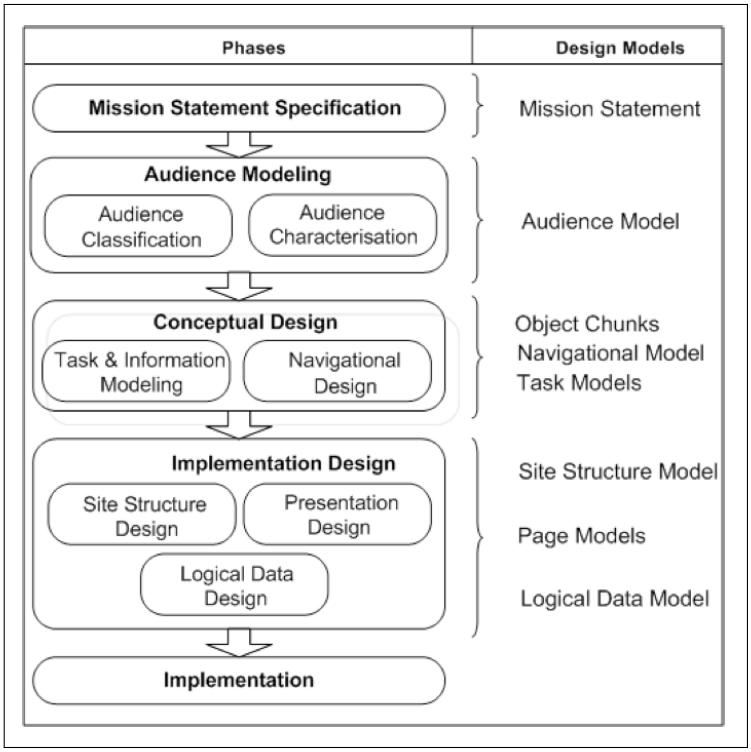


Figura 2.4 – WSDM: Procés Desenvolupament

un que es basa en les activitats de l'organització (es determinen els rols dels membres de l'organització) i l'altre que identifica els requeriments de funcionalitat d'informació i es crea una matriu d'usuaris-funcionalitat.

Després, cal caracteritzar l'audiència determinada. Per això, es defineixen les característiques rellevants per a cada classe d'audiència, com ara el rang d'edad, idioma, nivell d'experiència, etc. Aquestes característiques seran traduïdes en fases posteriors en requeriments d'usabilitat, i en la fase d'implementació com a requeriments de presentació que defineixen el "*look and feel*" de l'aplicació.

Els requeriments informals definits en el modelatge de l'audiència són transformats en descripcions formals que s'empren per generar el

sistema Web. Aquesta tasca es duu a term en la fase de **Modelatge Conceptual**, que es divideix en dos pasos: 1) modelatge de tasques i informació, i 2) modelatge navegacional.

Enlloc de començar el procés de modelatge mitjançant un model de dades o del domini, en WSDM s'analitzen els requeriments de cada una de les classes de l'audiència detectades. L'objectiu del model de tasques és el de detallar les diferents tasques que cada membre d'una classe d'audiència deu ser capaç de dur a terme, a més de descriure formalment la informació i funcionalitat requerida per a cada tasca.

D'aquesta manera, cada requeriment definit en una classe d'audiència és traslladat a una tasca que satisfà aquest requeriment. Cada tasca és modelada emprant una notació CTT ("*Concurrent Task Tree*") [96], modificada lleugerament per adequar-la a les particularitats de les aplicacions Web.

Quan el model de tasques ha sigut completat, un "*object chunk*" és creat per cada tasca elemental del model. La funció d'aquestos objectes és de definir formalment la informació i funcionalitat associada a una tasca. WSDM empra OWL com llenguatge per especificar aquestos "*object chunks*", per facilitar la integració amb ontologies existents. Per definir la funcionalitat, es proporciona un llenguatge de manipulació d'informació basat en una notació gràfica, que permet representar requeriments funcionals de certa complexitat. També permet la incorporació de serveis Web externs, per ser emprats com components funcionals del sistema.

L'objectiu del Disseny Navegacional és definir l'estructura conceptual del lloc Web i modelar com els membres poden navegar pel sistema per realitzar les seues tasques. Per a cada classe d'audiència, es crea un "*navigation track*", que es deriva a partir del model de tasques de la classe d'audiència corresponent. A continuació, tots els "*tracks*" són combinats en una estrucutra navegacional bàsica mitjançant enllaços d'estructura.

Després, s'afegeixen els enllaços semàntics entre els objectes del domini i els enllaços navegacionals de suport que faciliten la navegació (enllaç a la pàgina principal, dreceres, etc.). El resultat és un model navegacional format per components on la informació s'agrupa en els components (mitjançant els "*object chunks*") i enllaços entre components (definint aspectes navegacionals).

La següent fase del mètode és el **Disseny de la Implementació**. L'objectiu d'aquest model és completar el disseny conceptual amb detalls necessaris per a la implementació. Si el sistema Web és generara directament a partir del model conceptual, la aplicació generada resultaria molt simple a nivell de presentació. En aquesta fase es realitzen tres tasques:

1. *Disseny de l'estructura del lloc Web*: en aquesta etapa el dissenyador decideix com agrupar les pàgines en components del model navegacional. Aquesta agrupació es realitza tenint en compte les diferents classes d'audiència, que indiquen els components que deuen ser agrupats junts en funció de la informació presentada. Encara que per defecte un component s'agrupa en una única pàgina, és possible emprar diferents pàgines per representar un únic component. El resultat d'aquesta etapa és un model d'estructura del lloc web, que representa gràficament les pàgines (abstractes) i components.

2. *Disseny de la Presentació*: defineix l'aparença visual del sistema Web. Per facilitar aquesta tasca, es defineixen plantilles associades a un tipus de pàgina (pàgina principal, pàgina d'informació, etc.). Per al disseny d'aquestes plantilles es proposen una sèrie de conceptes de presentació per determinar el posicionament dels components, especificar enllaços, mostrar informació multimedia, crear formularis, definir capçaleres/peus de pàgina, ..., unit a l'ús de plantilles CSS. El resultat d'aquesta etapa és un model

de presentació compost per un conjunt de plantilles i estils per a cada pàgina del model d'estructura.

3. *Disseny Lògic de la Informació*: a partir dels "*object chunks*" i la seua descripció semàntica mitjançant una ontologia de referència, es deriva l'estructura d'informació del sistema. Si no existeix una capa de persistència per a la informació disponible, aquest esquema es tradueix en un esquema lògic de dades que done suport a la informació (com en una base de dades relacional), un esquema XML o un esquema semàntic expressat en RDF/OWL. Si existeix el sistema d'emmagatzemament a priori, cal definir les correspondències entre els conceptes de l'ontologia i el sistema d'emmagatzematge.

Per últim, cal realitzar la **Implementació** del sistema Web. El mètode defineix una sèrie de transformacions que tradueixen els conceptes dels models anteriors en una aplicació Web. Malgrat açò, actualment no existeix una eina que done un suport a aquesta etapa, encara que hi ha definides amb XSLT les transformacions com a prova de concepte.

En el procés d'implementació també s'aprofita per a fer anotacions semàntiques al codi obtés, per poder després emprar tecnologies semàntiques i llenguatges de consulta semàntica.

Com a aspectes possitius d'aquesta aproximació és la forta presència de l'audiència (usuaris) en tot el procés de desenvolupament. A més a més, ha incorporat una topologia prou completa de tipus d'enllaços navegacionals a nivell de model.

Com a aspectes a millorar, la necessitat d'un suport sistematiztat de producció de les aplicacions finals i un millor tractament de la dinàmica del sistema (operacions funcionals).

2.5.5 UWE

"*UML-based Web Engineering*" (UWE) [65, 54] és una metodologia per al desenvolupament de sistemes Web que es caracteritza per: 1) l'ús estricte de l'estàndar UML per definir els seus models conceptuals i 2) el suport al Disseny i Implementació de un Entorn MDA per a la obtenció de les aplicacions Web, mitjançant una eina de codi obert, ArgoUWE [63].

UWE pot considerar-se "UML compliant" ja que els seus elements de modelatge estan definits com extensions al metamodel d'UML. Si bé UML no és allò suficientment expressiu per modelar els requeriments dels sistemes Web, UWE introdueix elements específics del domini Web mitjançant els mecanismes d'extensió d'UML ("*profiles i restriccions OCL*") d'una manera conservativa-incremental[3]. Amb açò s'aconsegueix ser compatible amb l'estàndar *MOF* [87], de manera que es poden exportar els models a XMI i ser intercanviats amb altres eines. El fet de seguir estàndars li obre la porta a ser emprat amb altres eines.

L'especificació d'un sistema Web amb UWE involucra la definició de cinc models: *Modelatge de Requeriments*, *Model de Contingut*, *Model Navegacional*, *Model de Processos*, *Model de Presentació* i *Model d'Adaptativitat*.

En primer lloc es troba el **Model de Requeriments**. Aquestos s'especifiquen a dos nivells de detall o granularitat. En primer lloc es realitza una descripció de la funcionalitat mitjançant Casos d'Ús UML. En distingeix de tres tipus: 1) de *navegació*, que modelen interaccions amb l'usuari; 2) de *procés*, que descriuen les tasques de negoci que els usuaris han de dur a terme; i 3) de *personalització*, emprats per especificar els requeriments d'adaptació de l'aplicació als usuaris. En segon lloc es realitza una descripció més detallada de cada cas d'ús detectat mitjançant Diagrames d'Activitat d'UML. D'aquesta manera es

[3]Tot element de modelatge d'UWE hereta d'almenys un element del metamodel d'UML.

representa el fluix d'accions que el cas d'ús i els usuaris han de realitzar per a dur a terme una tasca.

Després dels requeriments, es defineix el **Model de Contingut**. L'objectiu d'aquest model és de proporcionar una especificació visual, basada en UML, de la informació rellevant en el domini del sistema Web. Aquesta informació és la que defineix el contingut que serà mostrat a l'usuari. Aquest model també pot contenir entitats emprades per a introduir aspectes de personalització.

El **Model Navegacional** descriu l'estructura de navegació del sistema Web. Aquest model es composa fonamentalment de classes navegacionals que representen els nodes de l'aplicació. Aquestos nodes poden ser alcançats mitjançant enllaços de navegació que conecten nodes entre sí. Per tal que l'usuari puga accedir a la lògica de negoci de l'aplicació, els processos de negoci subjacents del sistema deuen incorporar-se dins l'estructura navegacional. Aquestos processos són derivats a partir dels casos d'ús de tipus *procés*. Els punts d'entrada i d'eixida a aquestos processos es modela mitjançant *classes de procés* ("*process class*"), i aquestes classes poden relacionar-se entre sí mitjançant *enllaços de procés* ("*process links*").

Cada procés inclós al model navegacional ha de ser refinat mitjançant el **Model de Processos**. Aquest model especifica els processos com Diagrames d'Activitat UML que descriuen el fluix del procés, les accions possibles, les alternatives i les crides a la lògica de negoci.

Per últim, cal introduir certs aspectes més relacionats amb l'interfície, mitjançant el **Model de Presentació**. Aquest model proporciona una vista abstracta de la interfície a partir del model navegacional. Aquest model descriu l'estructura bàsica de la interfície i quins elements gràfics s'empraran per representar els nodes navegacionals. Els elements bàsics d'aquest model són les *classes de presentació*, que s'asocien als elements del model navegacional (classes navegacionals i de procés, primitives

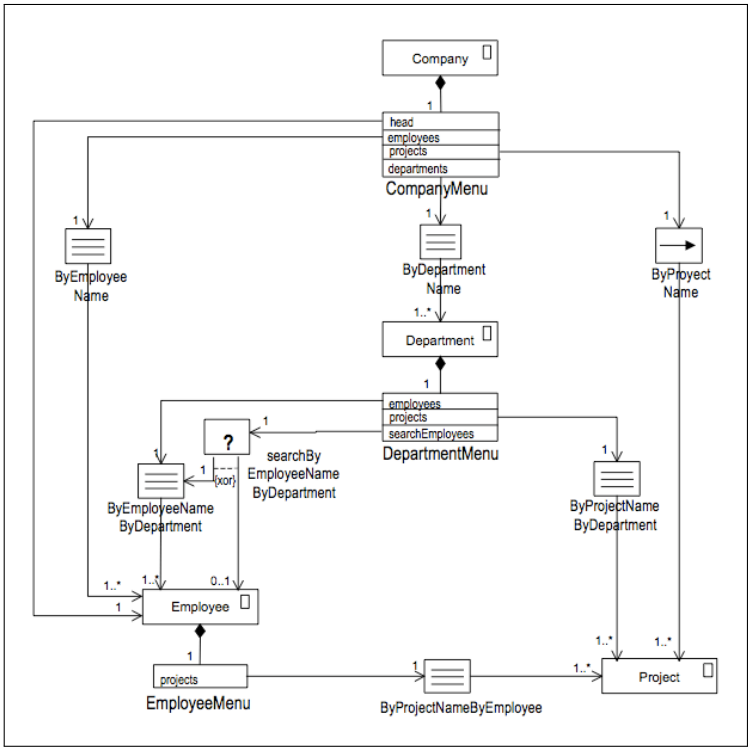


Figura 2.5 – UWE: Exemple Estructura Navegacional

d'accés, etc.). Una classe de presentació es composa d'elements d'interfície com ara text, enllaços, formularis, botons, images, etc.

Generalment, la informació de varios nodes navegacionals es presenta a l'usuari en una única pàgina web. El concepte de pàgina web es representa en UWE mitjançant la primitiva *page*, que actua com contenidora de classes de presentació i grups de presentació (agrupacions recursives de classes i grups de presentació).

Per tractar l'adaptativitat, UWE ho fa de manera diferent a la resta de models. El **Model d'Adaptativitat** es defineix de manera no intrusiva a la resta de models, emprant tècniques de modelatge

orientat a aspectes [68]. S'introdueix la primitiva "aspecte", composta d'un "*pointcut*" i un "*advice*". El "*pointcut*" és un patró parametritzable que selecciona els elements del model original amb els que coincideix ("*matching*"). Per a cadascun d'aquestos elements, l'"*advice*" afegeix una nova característica (per exemple una nova relació o un nou atribut a cadascun dels elements seleccionats. D'aquesta manera és possible definir un "*pointcut*" que detecte totes les classes que pertanyen a un tipus d'usuari i associar a través de l'"*advice*", un atribut específic per a aquest tipus d'usuari. En l'orientació a aspectes, açò es coneix com *weaving*.

A més de definir, com s'ha comentat, una extensió UML, UWE es caracteritza per definir un procés de desenvolupament a partir d'aquestos models. Aquest procés està basat en l'aplicació dels principis de MDA, emprant altres estàndars de la OMG com ara MOF, UML, OCL i XMI. El procés es basa en una sèrie de transformacions successives a través de les quals s'obté l'aplicació Web [75]. UWE situa el model de requeriments al nivell CIM. La primera transformació consisteix en traslladar aquestos requeriments a models funcionals mitjaçant regles QVT, obtenint una primera versió del PIM de l'aplicació (basat en el metamodel definit en MOF). El segon pas consisteix en realitzar un refinament d'aquestos models marcant les classes del model de contingut rellevant des del punt de vista de la navegació. Amb aquest refinament és possible derivar un primer model navegacional aplicant un conjunt de regles predefinides en OCL. Per últim, es realitza un procés d'integració en el que els models funcionals són integrats en un únic model, i després s'integra amb un altre model que introdueix aspectes de disseny arquitectònic de l'aplicació.

Com resultat s'obté el **Model d'Integració** que cubreix tant els aspectes funcionals com arquitectònics. A partir d'aquest model, i realitzant transformacions en ATL, s'obtenen els models específics de plataforma (PSM), que són emprats com punt de partida per a la

generació de codi.

Com a virtuts de l'aproximació està la orientació cap a entorns adaptatius (ha incorporat algunes idees del camp de la Hipermedia Adaptativa) i la seua definició rigorosa del procés de producció, amb la introducció de nombroses guies d'ajuda al modelatge. A més a més, l'utilització d'estàndars notacionals li confereixen un major àmbit d'impacte, potenciat per la creació de l'eina CASE.

Com a punts a millorar, continua essent necessari tenir més en compte el model dinàmic, en el qual actualment estan treballant. D'altra banda, l'eina CASE dóna un suport limitat únicament al modelatge conceptual.

2.5.6 Hera

El propòsit del mètode Hera [55, 139] és el de suportar el disseny d'aplicacions web adaptables i personalitzables, seguint una estructura navegacional definida sobre informació estructurada semànticament.

El punt de partida en Hera és el **Model d'Informació** a través de la construcció de dos models conceptuals: el *Model de Domini* i el *Model de Contexte*. El Model de Domini s'encarrega de deescriure l'estructura d'informació, especificant l'estructura semàntica del contingut. Aquesta especificació és definida emprant l'ontologia OWL, a través d'un editor genèric d'ontologies com *Protégé* [125].

D'altra banda, el *Model de Contexte* apareix per donar suport a l'adaptació de l'aplicació Web en base al contexte de l'usuari. Aquest model s'encarrega de definir la informació que serà emprada per crear els continguts adaptats i personalitzats. Hera distingeix tres tipus d'informació de contexte: de *sessió*, d'*usuari* i *global*, en funció de l'àmbit d'aplicació. Ambdós models (de Domini i de Contexte) són implementats a través del repositori ontològic *Sesame* [138], que emmagatzema els models en RDF [143].

Una vegada descrita l'estructura d'informació del sistema, el següent

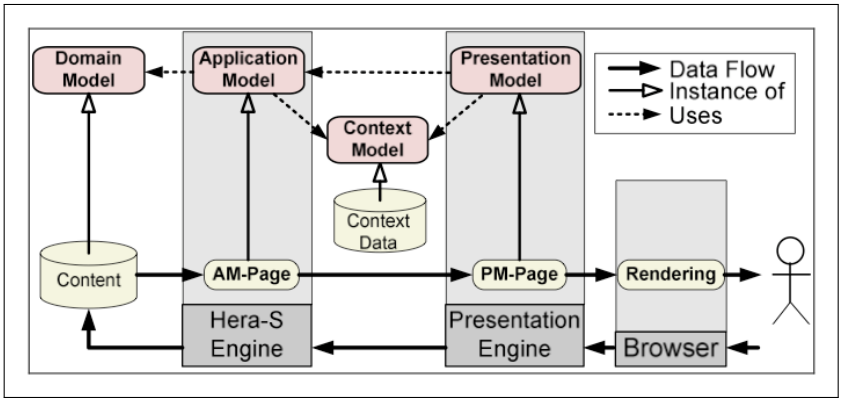


Figura 2.6 – Hera: Procés de Desenvolupament

pas consisteix en definir la pròpia aplicació. El **Model d'Aplicació** especifica el comportament navegacional, permetent establir quina informació és mostrada a l'usuari i cap a quines pàgines web pot navegar per obtenir-la. Aquest model és a més a més dinàmic, de manera que l'accés a la informació pot ser adaptada a un usuari i contexte específics.

La instanciació del mode l'aplicació en base als models anteriors dóna lloc al *Model de Pàgines de l'Aplicació* (AMP), que estan formades pel contingut a mostrar i enllaços navegacionals a altres AMPs. Per tant, la principial funció d'aquest model és la d'accedir a la informació dels models de domini i contexte. Ja que aquesta informació es troba definida en RDF, el model d'aplicació defineix aquest accés a través del llenguatge SeRQL (Sesame RDF Query Language), un llenguatge de consulta i transformació RDF. Aquest llenguatge empra plantilles basades en grafs a partir de les quals instància les variables a valors que compleixen el graf RDF buscat. La seua sintaxi propera a SQL la fa especialment expressiva en el contexte d'aplicació d'Hera. En conseqüència, la resta d'elements conceptuals del model d'aplicació són construïts en base a aquest llenguatge, i per tant en un entorn que segueix els principis de la web semàntica [14].

Els constructors principals del model d'aplicació són les *unitats navegacionals* i les *relacions*. La instanciació d'unitats i relacions es realitza mitjançant consultes SeRQL sobre la informació de domini i contexte. En aquestes consultes pot definir-se l'adaptativitat d'una *unitat*, establint condicions sobre el contexte (per exemple, l'usuari conectat) que defineixen quina informació del domini es mostra en funció de si s'acompleixen o no. Una unitat pot ser emprada per representar una *pàgina*, que agrupa distints elements que seran mostrats a l'usuari. Aquestos elements són anomenats *atributs* i representen una porció elemental d'informació basada en el model de domini. El seu valor ve definit per una consulta en SeRQL. Les *relacions* s'encarreguen d'enllaçar unitats entre sí. En Hera és possible emprar les relacions tant per definir unitats que es troben contingudes en altres, com per especificar navegació entre unitats.

El model d'aplicació inclou a més una sèrie de constructors addicionals per construir aplicacions més complexes:

- *Update Queries*: permeten actualitzar i/o modificar la informació del model de contexte en temps d'execució.
- *Frame-Based Navigation*: quan una unitat es troba continguda en una altra, la navegació per defecte implica reemplaçar completament la unitat contenidora per la continguda. Aquest mecanisme permet definir la navegació de manera que únicament es reemplaça la unitat continguda, mantenint la unitat contenidora original[4].
- *Forms*: aquesta unitat extèn una unitat genèrica definint una colecció d'elements d'entrada i una acció que serà executada quan el formulari s'envie al servidor.
- *Scripting Objects*: permet al dissenyador especificar un script definit en un llenguatge extern (JavaScript, VBScript, etc.) que no serà processat pel motor de presentació.

[4] Aquest comportament és semblant al constructor *frame* d'HTML.

- *Service Objects*: permeten afegir funcionalitat que s'obté a través d'un servei extern a l'aplicació, com per exemple un Servei Web.

Per últim, és necessari especificar com la informació serà finalment presentada a l'usuari. El **Model de Presentació** s'encarrega d'especificar com el contingut de les unitats navegacionals s'han de mostrar. Aquest model es defineix en base a *regions* i relacions entre aquestes. Una regió és una abstracció que representa una àrea de la pantalla de visualització. Cada regió agrupa varies unitats del model d'aplicació, i a la seua vegada, poden contenir altres regions de manera recursiva. A més a més, aquest model especifica la disposició ("*layout*") i l'estil de les diferents regions a través de flls d'estil CSS. La implementació del model de presentació es realitza prenent com base el model de documents basat en components *AMACONT* [7].

Aquesta aproximació defineix la implementació d'aplicacions Web personalitzades com l'agregació de diferents components documentals. A aquestos components se'ls pot associar una descripció de *presentació abstracta*, basada en el model de presentació, de manera que després puguen ser transformats a diferents tecnologies Web.

2.5.7 OOH

El mètode d'Hipermèdia Orientat a Objectes (OO-H) [92][52], a partir de la funcionalitat i la lògica que oferisca algun model d'estructura d'informació (inicialment OO-Method).

OO-H defineix un model de processos genèric que ofereix als dissenyadors la semàntica y notació necessària per al desenvolupament d'interfícies basades en el Web i al mateix temps ofereix la possibilitat de conectarles amb mòduls de lògica d'aplicació preexistents, facilitant la migració d'aplicació legades. OO-H agrega dos diagrames complementaris: el *Diagrama d'accés navegacional (NAD)*, que defineix la vista navegacional; i el *Diagrama abstracte de presentació (APD)*, que captura els conceptes

relatius a la presentació. Agregant a aquestos un *Catàleg predefinit de patrons d'interfície hipermedial* s'obté la informació necessària per al disseny de la interfície. Ofereix, un *Compilador de models* que genera la interfície per a la plataforma client indicada.

Existeix un model de procés que compren les fases que el dissenyador cubreix per obtenir i construir la interfície funcional que satisfaga els requeriments de l'usuari: 1) Partint del diagrama de classes d'estructura d'informació, es defineix per a cada tipus d'usuari un NAD, reflexant la informació, serveis i rutes de navegació que satisfan els requeriments. 2) Aplicant uns passos definits pel mapeig NAD2APD, es genera un interfície Web per defecte, que serà la primera aproximació a la interfície Web final i que requerirà refinaments. Aquesta interfíce es modela amb un APD emprant una estratègia de plantilles. 3) Per tal de millorar la qualitat de la interfície, el APD per defecte es refina aplicant un o més patrons del Catàleg. És possible tenir un o més APD per a cada NAD, mostrant diferents maneres per visualitzar els mateixos requeriments de navegació. 4) Finalment, una vegada refinat l'APD, és possible generar la interfície de l'aplicació Web per a l'ambient desitjat, recolçant-se amb tecnologies com ara HTML, WML, ASP, JSP, etc.

2.6 Eines de Suport al Desenvolupament Web

En aquesta secció s'analitza l'estat de l'art des de l'àmbit pràctic i industrial del desenvolupament d'aplicacions Web seguint un enfoc DSDM. En aquest entorn, anem a difereciar entre les eines que provenen del món acadèmic de les industrials, sempre en l'àmbit del desenvolupament Web.

En aquest estudi apareixen les més rellevants o interessants en certs aspectes. Tanmateix, en [32] pot conseguir-se informació més completa sobre les eines disponibles al mercat.

2.6.1 HyperDE

HyperDE [83] és un entorn DSDM basat en un framework MVC extés mitjançant primitives navegacionals. Aquesta eina permet al dissenyador introduir els diferents models OOHDM i la definició d'una interfície des d'el punt de vista del patró MVC. Amb aquesta informació, es genera l'aplicació Web. Com part de l'aplicació generada, incorpora aspectes funcionals bàsics per crear i modificar informació al sistema.

En la versió actual de HyperDE, s'empra la variant SHDM [71] de OODHM, la qual empra un model d'objectes derivat d'un model expressat en RDF, capaç de descriure semànticament dades i meta-dades.

La implementació de HyperDE pren com punt de partida el framework *Ruby on Rails* [113], al que se li ha modificat la capa de persistència per altra basada en la base de dades Sesame RDF [109]. Tant els metamodels SHDM com la informació del sistema són emmagatzemats en aquesta capa com informació RDF. Tota la funcionalitat de HyperDE està disponible a través d'una interfície web que permet manipular en temps d'execució la informació del sistema. Així mateix, es proporciona un llenguatge específic (DSL) com extensió de Ruby, que permet la manipulació directa mitjançant scripts del model de l'aplicació i del metamodel.

D'altra banda, la funcionalitat de l'aplicació és implementada mitjançant operacions associades a les diferents classes navegacionals. El codi per implementar aquestes operacions empra el DSL abans comentat, que permet definir tant operacions d'actualització del model de dades com crear o alterar els valors dels objectes navegacionals.

Per obtenir la població d'un contexte, HyperDE defineix un llenguatge de consulta simplificat, basat en l'especificació de la classe origen i destí, la relació i un criteri d'ordenació. Tanmateix, també suporta la definició de consultes més complexes mitjançant RQL o amb expressions Ruby sobre el DSL propost.

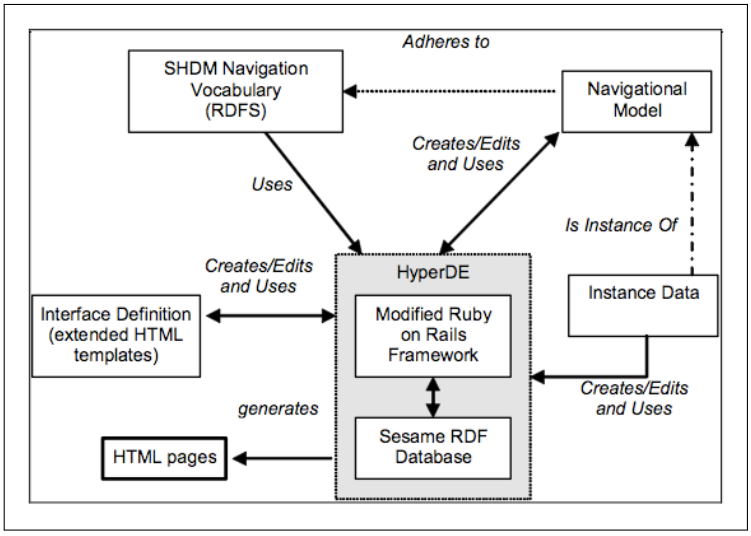


Figura 2.7 – HyperDE: Estructura de l'eina

L'entorn genera per defecte una interfície bàsique per donar suport a l'aplicació. A partir d'aquesta interfície inicial, el dissenyador té la llibertat d'afegir aspectes de presentació més complexes. La personalització de la interfície permet inclús la definició d'una distribució concreta per un node o una classe navegacional.

Per últim, cap ressenyar que existeix una versió de l'entorn que permet la generació de codi d'una manera similar per a la plataforma .NET.

2.6.2 WebRatio

El desenvolupamnet d'aplicacions basat en WebML es duu a terme mitjançant l'eina **WebRatio** [146]. L'arquitectura d'aquesta eina es divideix en dues capes: la *Capa de Disseny*, que proporciona funcions per a l'edició visual de les especificacions, i la *Capa d'Execució*, que implementa els serveis necessaris per a l'execució de les distintes unitats

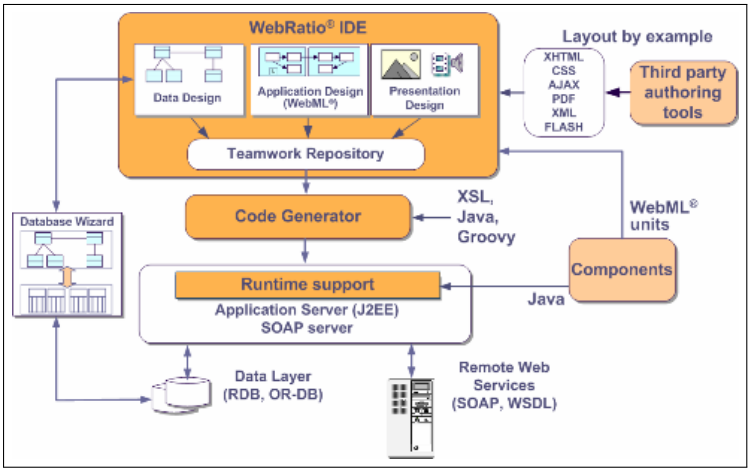


Figura 2.8 – WebRatio: Arquitectura

WebML en una aplicació.

La *Capa de Disseny* inclou una interfície d'usuari per al disseny dels mòduls de Dades i d'Hipertext, que són emmagatzemats com XMl. D'altra banda, el mòdul *Database Synchronizer* s'encarrega de mapejar les entitats i relacions de l'esquema conceptual a una o varies fonts de dades preexistents o creades mitjançant l'eina. Aquest mòdul inclou a més un mecanisme d'enginyeria inversa que permet inferir l'esquema conceptual a partir d'una font de dades preexistent.

L'eina es composa d'un tercer mòdul, anomenat *EasyStyler Presentation Designer*, que proporciona la funcionalitat necessària per definir la presentació visual de l'aplicació. Aquest mòdul permet crear al dissenyador fulls d'estil en format XSL per associar-les a les pàgines definides en WebML i organitzar la distribució de les diferents unitats que formen cada pàgina.

La Capa de Disseny es troba conectada amb la *Capa d'Execució* mitjançant el generador de codi de WebRatio. Aquest generador empra transformacions en XSLT per traduir les especificacions emmagatzema-

des en XML en codi executable per a la plataforma J2EE . En concret, un conjunt de traductors XSL produeixen diverses plantilles dinàmiques de pàgina i descriptors d'unitat. Una *plantilla dinàmica de pàgina* expresa el contingut i la implementació d'una pàgina en un llenguatge Web (HTML, WML, etc.). En canvi, un *descriptor d'unitat* és un fitxer en XML que expresa les dependències d'una unitat WebML amb la capa de dades (per exemple, el nom de la taula que proporciona la població a la unitat).

Tant la capa de disseny i execució com el generador de codi estan articulats en una arquitectura basada en plug-ins. D'aquesta manera, es facilita la integració de nous components a través de descriptors XML i ser importats en la capa de disseny com unitats WebML personalitzades.

D'altra banda, el generador de codi pot ser extés mitjançant regles XSLT addicionals per produir el codi d'aquestos nous components. Aquesta arquitectura permet a WebRatio ser extesa en el futur amb noves funcionalitats.

2.6.3 Hera Suite

Al mètode Hera, els models s'especifiquen de forma textual emprant RDF. Aquest fet fa que la gestió d'aquestos models siga complicada i pot induir fàcilment a errors. L'editor **Hera Suite** [138] facilita aquesta tasca al dissenyador proporcionant un entorn gràfic per gestionar els models de *domini*, *context* i *aplicació*. Aquestos models poden ser exportats a una especificació pura en RDF, que després l'eina podrà altra vegada importar.

L'editor de Models de Domini permet definir tant classes com propietats de les classes (segons l'estructura d'organització que proposa la Web Semàntica [20]). Per a cada classe, és possible definir el tipus de dades i les seues propietats a partir d'un conjunt predefinit (*String*, *Image*, etc.), així com relacions d'herència. Si per alguna raó són necessaris constructors més complexes, el dissenyador sempre pot continuar emprant un

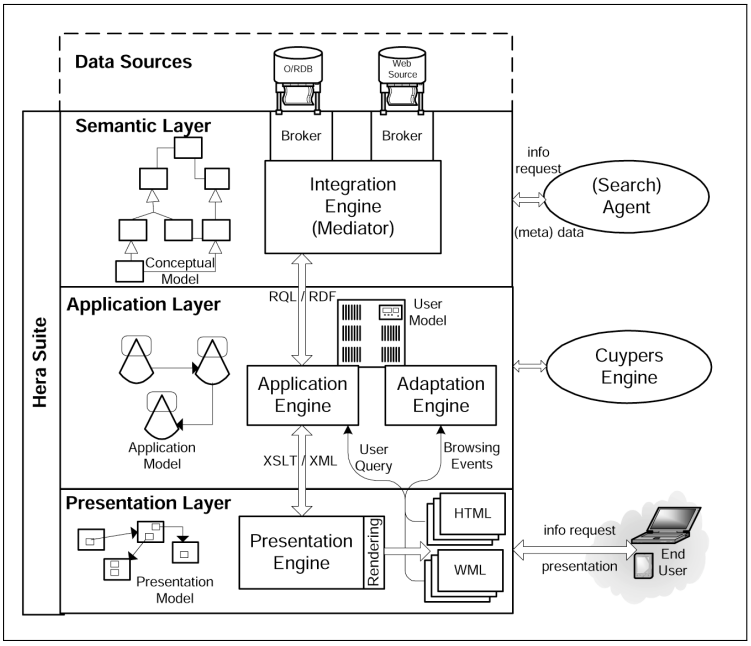


Figura 2.9 – Hera Suite

editor genèric per generar la sintaxi adequada.

L'editor de Models d'Aplicació permet especificar l'organització de les unitats, els seus elements i relacions. Per defecte les consultes que defineixen les unitats són ocultades en la vista gràfica, però poden ser configurades per a treballar en mode textual. L'editor proporciona ajudes per a la definició dels constructors més simples (tipus de dades) però si és necessari, el dissenyador pot definir les seues pròpies consultes i propietats. Tanmateix, l'editor actual encara no suporta tots els constructors disponibles en Hera.

La implementació dels models es realitza a través d'un motor que genera una vista hipermedial en funció de l'especificació del model de l'aplicació. Aquest motor, anomenant Hera-S [138], està basat en un implementació prèvia anomenada HPG-Java [45]. Aquest motor es basa

en un component servidor, implementat com un Servlet de Java que per a cada petició de pàgina, genera el contingut de manera dinàmica tenint en compte l'adaptació necessària. El motor actual Hera-S té un funcionament semblant però atenent als nous models definits a traves de consultes SeRQL. Si bé la implementació continua sent en forma de Servlet, l'emmagatzemament i manipulació dels models es realitza a través de respositoris *Sesame* que suporten qualsevol font d'informació RDF.

El motor està definit en base a una arquitectura orientada a events. Quan una petició de pàgina és rebuda pel *Component Servidor de Presentació*, la petició és traduïda a un petició d'una *AMP* (Secció 2.5.6). Simultàniament, els components d'usuari i sessió actualitzen la informació de context al repositori. A continuació, el component *AMPGen* recupera la part del model d'aplicació que conté l'especificació necessària per a la construcció de la AMP. Aquesta AMP és implementada internament al repositori Sesame, de manera que totes les seues operacions de transformació es duen a terme mitjançant consultes SeRQL. Quan la AMP ha sigut finalment construida, és enviada cap al Component Servidor de Presentació, que s'encarrega de transformar-la en un pàgina que puga ser visualitzada per un navegador. Per aconseguir-ho, es recolça en la creació de documents AMACONT [7], que després són transformats a un format d'eixida adequat en funció del client (XHTML, cHTML, WML, ...), adaptant-se als seus requeriments.

2.6.4 AndroMDA

AndroMDA [9] és un framework MDA per a la definició de generadors de codi. A partir de models UML, és capaç de generar un conjunt e components funcionals en una plataforma específica (J2EE, Spring, .NET). AndroMDA proposa una arquitectura modular basada en diferents plugins que poden ser intercanviats i redefinits per adaptar-se

a les necessitats d'un procés MDA específic. El plugin principal en el que AndroMDA es sustenta és el **cartutx**, que empaqueta un motor de transformacions per a una plataforma tecnològica concreta.

El procés de generació de codi d'AndroMDA està basat en l'ús d'estereotips UML, de manera que el motor analitza el model buscant classes marcades amb un estereotip específic. Quan un estereotip és reconegut, s'invoca al cartutx corresponent que s'encarregarà de generar el codi per a aqueixa classe. Per poder generar el codi, el cartuxt inclou un conjunt de plantilles de text, que són completades en funció de la informació obtesa de la classe.

Actualment AndroMDA suporta *Velocity* [10] i *FreeMarker* [47] com llenguatges de plantilles. Per defecte, AndroMDA inclou una sèrie de cartutxos predefinits per a diferents plataformes i arquitectures, com ara EJB, Spring, Hibernate, Struts, JSF, Axis i jBPM. Tanmateix, l'usuari pot, si vol, definir el seu propi cartutx, indicant com han de ser processats els models i plantilles de generació de codi associades.

El framework no proposa una eina CASE específica per a la definició dels models d'entrada. En canvi, proporciona suport complet per al metamodel UML 1.4 [84] (i en un futur per a la versió 2.0) i per a metamodels definits mitjançant l'estàndar MOF de la OMG. D'aquesta manera, és possible definir models emprant eines compatibles amb UML, com ara *MagicDraw* [82], *Poseidon* [51] o *Enterprise Architect* [124].

Per garantitzar la correcció dels models, AndroMDA incorpora un mecanisme de validació basat en restriccions OCL associades a les classes el metamodel. Per defecte inclou una sèrie de restriccions per evitar els errors de modelatge més habitual. No obstant, és possible definir restriccions per a un odel específic. AndroMDA planeja donar suport a models definits amb EMF (*Eclipse Modelling Framework*) [40] i transformacions model-a-model amb *ATL* [12].

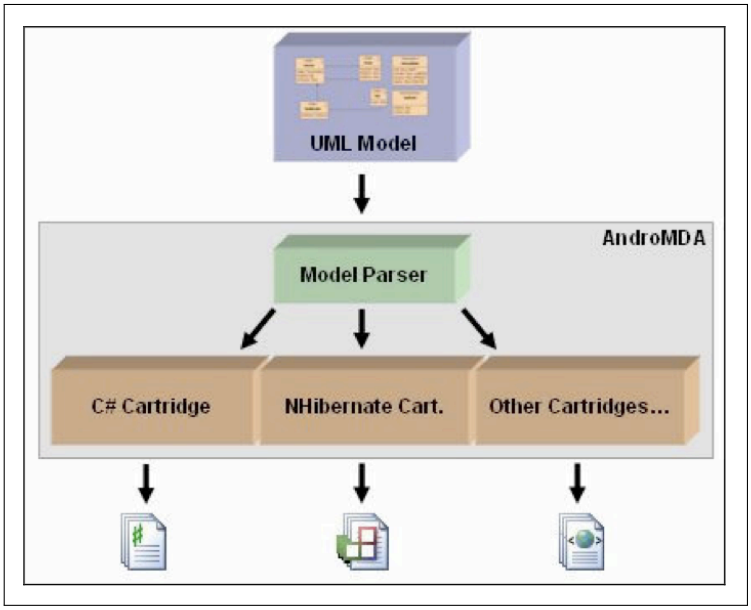


Figura 2.10 – AndroMDA: Arquitectura

2.6.5 OptimalJ

OptimalJ [33] és un entorn MDA per al disseny, desenvolupament i distribució d'aplicacions J2EE. Aquest entorn fou un dels pioners a l'hora d'adoptar l'estàndar MDA. Seguint aquestos principis, OptimalJ proporciona una clara distinció entre els *models del domini* i els *models dels components de l'aplicació*.

Per a aconseguir-ho, els models es divideixen en tres nivells d'abstracció: 1) el *Model de Domini*, que defineix el domini de negoci sense tenir en compte detalls relatius a la plataforma, 2) el *Model d'Aplicació*, que especifica l'aplicació en base a certa tecnologia però sense detallar aspectes relacionats amb el codi, i 3) el *Model de Codi*, que defineix l'aplicació com codi que pot ser compilat i executat. A partir d'aquestos models, que poden ser dissenyats per distints responsables, és possible

generar tant les classes Java com els scripts SQL de l'aplicació.

En OptimalJ els diferents nivells de modelatge es troben relacionats mitjançant patrons, de manera que els canvis realitzats en un nivell superior són propagats a nivells inferiors. La generació dels models és incremental, poden modificar els models orige i destí en paral·lel sense sobreescriure els canvis realitzats al model destí. Aquest mecanisme garantitza la sincronització entre els models. A més a més, els patrons aplicats per OptimalJ poden ser configurats i desactivats per evitar la regeneració de certs elements del model destí.

L'arquitectura d'una aplicació desenvolupada mitjançant OptimalJ es divideix en tres parts: 1) un *framework genèric* d'aplicacions de caràcter estàtic, predefinid d'avantmà; 2) un *framework dinàmic* de l'aplicació que és generat a partir del Model d'Aplicació; i 3) *codi* que s'introdueix a mà pel desenvolupador.

Actualment, OptimalJ suporta diferents frameworks que poden ser emprats com part estàtica de l'aplicació. En concret, per a la *capa de presentació*, és possible emprar *Apache Struts* [126], mentre que per a la resta de capes es proposa l'ús del framework *Altura* com a primera elecció. Aquest framework proporciona implementacions de patrons genèrics que faciliten l'implementació d'aplicacions. D'altra banda, el framework dinàmic d'aplicació és específic de l'aplicació a generar, ja que s'obté a partir dels models conceptuals d'aquesta. El framework dinàmic empra els frameworks estàtics definits prèviament i genera un conjunt de regions de codi no editables anomenada "*guarded blocks*".

Per últim, la lògica de negoci que no és suportada pels models deu ser implementada manualment. Aquest codi pot ser insertat al framework d'aplicació mitjançant regions editables, anomenades "*free blocks*". En aquestes regions es poden emprar tant les classes del framework estàtic com el dinàmic.

2.6.6 ArcStyler

ArcStyler [60] és una eina de producció de programari que segueix la filosofia MDA i que està basada en dos pilars fonamentals: 1) un *editor de models UML*, i 2) un *motor de transformacions* mitjançant cartutxos MDA.

En primer lloc, l'Editor de Models UML és el punt de partida en el procés de desenvolupament de ArcStyler, proporcionant les característiques habituals que poden trobar-se en altres eines. Respecte a OMG, suporta la definició de *perfils UML*, la definició de diagrames personalitzats i mecanismes per documentar i validar els models de manera exhaustiva.

Tots els models generats mitjançant aquest editor són emmagatzemats en format XMI, podent importar-ne i exportar-los a *Rational Rose* [56]. L'editor també emfatiza la reutilització de models introduint mecanismes de composició entre models. D'aquesta manera, qualsevol canvi realitzat en un submodel, queda automàticament reflexat en el model compost. Tanmateix, el concepte que realment diferència a l'editor UML d'ArcStyler respecte al d'altres eines, són les *MDA Marks*. Aquestes marques són anotacions sobre elements del model que contenen informació referent a la plataforma d'implementació. Ja que aquestes marques poden ser afegides i eliminades fàcilment, permeten no haver d'introduir en el model conceptes depenents de la implementació. És més, distints conjunts de marques poden definir-se sobre un mateix model de manera que és possible generar codi per a diferents plataformes sense haver de realitzar modificacions al model inicial.

El motor de generació de codi de ArcStyler es basa en un conjunt de *cartutxos MDA* intercanviables i extensibles. Cada cartutx conté tots els elements i mecanismes específics d'una tecnologia concreta per transformar de manera òptima els models a un producte de programari. Emprant diferents cartutxos, ArcStyler suporta un ampli rang de plataformes, com

Java/J2EE (Java2, EJB 1.1, EJB 2.0, BEA WebLogic, IBM WebSphere i JBOSS) ó .NET (C# i ASP .NET).

En l'àmbit de les aplicacions Web és possible generar de manera automàtica "*fornt-ends*", interfícies Web, Serveis Web o Interfícies Externes (EAI) per a diversos tipus de plataformes. Per al modelatge d'una aplicació Web, ArcStyler empra els diagrames de classes, els diagrames d'activitat i un perfil UML. A partir d'aquestos models, és possible generar una aplicació Web que compleix amb el patró MVC, que suporta l'accés autenticat d'usuaris i amb la infraestructura necessària per a realitzar operacions CRUD.

A pesar que els cartutxos MDA de ArcStyler han sigut dissenyats tenint en ment un ampli espectre d'entorns, per a un projecte concret pot esdevenidre que no cubrisca totes les necessitats. Per solventar aquesta problemàtica, els cartutxos foren dissenyats per a l'extensió i personalització, a través de la definició de punts d'extensió que tot cartutx té definit, per poder ser modificat de manera no intrusiva conservant la funcionalitat prèvia.

A més a més, el desenvolupament d'aquestos cartutxos personalitzats també es realitza seguint la filosofia MDA. La transformació és modelada de manera visual mitjançant un profile UML específic i un cartutx especial s'encarrega de generar gran part del codi que implementa l'adaptació del cartutx a modificar. En la creació de nous cartuxtos des de zero s'aplica aquest mateix principi. En aquest cas, és possible emprar l'IDE de desenvolupament de cartutxos, que incorpora mecanismes per definir de manera visual la transformació del cartutx, depurar les regles de transformació i refinar de manera manual el codi a generar.

Per últim, cal destacar altres característiques interessants d'aquesta eina MDA. Si bé, ArcStyler enfatiça un procés de desenvolupament "*top-down*" en el que, a partir dels models, es genera el codi. El procés invers resulta si cap més interessant. ArcStyler proporciona el concepte de

Harvester, de manera que a partir d'una aplicació generada en J2EE és possible obtenir el model associat, aplicant enginyeria inversa. Aquest model pot resultar útil per tenir una visió global del sistema o per a ser traduït a una altra plataforma tecnològica.

Altre aspecte relacionat que incorpora ArcStyler són les transformacions model a model. L'eina incorpora un mòdul que permet definir aquest tipus de transformacions (per exemple, de processos de negoci a UML) i inclús definir "cadenes" que defineixen transformacions concatenades entre varios models.

2.7 Conclusions

Després d'analitzar cadascun dels mètodes, podem constatar similituts entre ells. En primer lloc, tots contenen un model d'*estructura d'informació o de domini* (normalment emprant el paradigma d'orientació a objectes), d'una manera prou consensuada. UWE i OOHDM proposen l'ús directament d'UML. WebML es basa en un model Entitat-Relació Extés, i tant WSDM com Hera empren notacions basades en la web semàntica, i en concret amb l'ús de llenguatges basats en RDF/OWL. Açò els permet l'ús d'ontologies existents, que els permet aprofitar de socarrel els avantatges que aquesta tecnologia introdueix: construcció ràpida de dominis amb dominis genèrics predefinits, compartició de models, consulta semàntica de models, etc. Tanmateix, el problema d'aquestes aproximacions, és que els llenguatges que proposa la web semàntica solen tenir una representació gràfica poc adient per al modelatge de sistemes d'un tamany considerable (poc compacta). Açò fa que els sistemes prompte creixen en tamany i dificulten el seu enteniment, a més les operacions de manipulació de la informació del sistema es realitza mitjançant RDF/OWL, que dóna la impressió d'estar "programant" enlloc d'especificant.

També existeix un consens total en la forta necessitat de modelar la *navegació*, i es sol construir sobre el model d'estructura d'informació (i manté relacions de dependència de recuperació d'informació). Només OOHDM com WSDM proposen primer la construcció del model navegacional, i derivar la informació del sistema a partir d'aquest.

A l'hora de modelar la navegació, també hi ha consens en el l'estructura subjacent: grafs dirigits. Aquestos grafs estan composats de nodes (que representen algun tipus de "contingut" en funció de cada aproximació) i enllaços entre nodes (que defineixen la navegació entre ells). Alguns fan una classificació detallada de tipus de nodes (com ara WebML o WSDM) i tots distingeixen tipus d'enllaços. En general, la semàntica és prou similar.

També apareix normalment un model abstracte de *presentació*, que descriu requeriments abstractes d'interfície, encara que les aproximacions difereixen en el nivell d'abstracció, i la manera en que es construeix. Així, en OOHDM i OOH primer es fa una descripció abstracta i després es detalla aquesta amb objectes concrets d'interfície. Es proposen tècniques i notacions adients.

Un altre model que també és habitual avuí en dia en aquest tipus d'aproximacions és el model de *personalització i/o adaptació*. En aquest model es representen requeriments d'adaptativitat del sistema, normalment a l'usuari (personalització). Alguns mètodes, com ara UWE o Hera tenen aquest aspecte com un pilar essencial dins el seu procés, i arriba a guiar la construcció dels sistemes.

Però, un aspecte que en general no està ben tractat és el de la descripció funcional del sistema. En la majoria dels mètodes, només es "declaren" les operacions funcionals del sistema (normalment com operacions de les classes del diagrama de classes), però no es fa una especificació abstracta del seu comportament. Tots els models que es presenten es refereixen fonamentalment a aspectes que donaran lloc a la

interfície de l'aplicació Web, les seues pàgines, menús, enllaços, etc.

De forma natural, també expresen la funcionalitat associada a la recuperació de la informacio, i alguns, operacions bàsiques sobre aquesta (operacions CRUD[5]). Tanmateix, no donen suport a l'especificació de funcionalitat més complexa, molt habitual en les aplicacions web (gestió d'un carret de la compra, pagament amb tarjetes de crèdit, etc.).

UWE i OOHDM, per emprar UML, proposen etiquetar l'especificació anotant el comportament, construir Diagrames de Seqüència (on s'ordenen les operacions) i emprar açò com a documentació per a la implementació del sistema. WebML, d'altra banda, sí que permet la especificació d'operacions funcionals, encara que per fer-ho cal emprar el llenguatge de consulta de dades SQL, fent depenent (i per tant perdent nivell d'abstracció) l'especificació del sistema en base a l'estructura de la Base de Dades. El fet de no realitzar una especificació completa de la funcionalitat, fa que les aplicacions Web generades (automàticament) a partir d'aquestos mètodes tinguen una manca greu de funcionalitat.

En general, les aproximacions parteixen d'una *especificació de requeriments* incial, de maneres diferents. Hi ha aproximacions que es basen directament en Casos d'Ús i Escenaris d'UML (com ara UWE) i altres que defineixen la seua pròpia notació (els UIDs de OOHDM o la Missió de l'Aplicació en WSDM). WebML no ho defineix explícitament, encara que podria empar-se'n alguna.

A l'hora de definir les *fases del procés de desenvolupament* és quan apareixen més diferències entre les aproximacions, i on es veu quines són les tasques dins el desenvolupament del sistema que tenen més rellevància. Tots enfaticen la necessitat del modelatge de l'estructura d'informació i de navegació, però no de la mateixa manera als models de presentació, de personalització o a l'especificació de requeriments.

Mentre que en UWE i en Hera es donen més detalls a l'hora de

[5]De l'angles "*Create, Relate, Update i Destroy*", Creació, Relació, Modificació i Destrucció, operacions bàsiques al món Orientat a Objectes.

definir els requeriments, en WebML i en OOHDM, sense obviar-los, no són enfatitzats al mateix nivell. Respecte a la presentació, tant Hera com WSDM fan ús de l'estàndar CSS (amplament acceptat en el disseny gràfic d'aplicacions web) introduint aspectes tecnològics a nivell de modelatge, mentre que la resta de mètodes aporten un model de presentació abstracte.

A pesar que anys enrere, la fase d'*implementació* era tinguda com menys rellevant en la majoria dels mètodes (només WebML i OOH apostaven per proposar una guia "fins el final"), avuí en dia, gràcies a l'aparició i assentament del Desenvolupament Dirigit per Models i MDA, quasi totes les aproximacions apliquen d'una o altra manera transformacions de models per aconseguir prototips del sistema.

Amb respecte a la *notació*, només UWE aposta clarament per UML, mentre que WebML i OOHDM empren un notació propietària, i marcadament diferenciada de UML. El fet d'emprar com notació UML es pot considerar un avantatge interessant per a UWE, a causa de la gran quantitat d'eines actualment en el mercat que donen suport a aquest llenguatge. Açò li confereix major "portabilitat" a altres eines. Malgrat açò, i a causa que els conceptes són semblants en totes les aproximacions, no seria molt complicat construir perfils UML de qualsevol aproximació per tal d'obtenir aquestos avantatges.

A més a més, l'expressivitat es considera suficient per als conceptes que es pretén capturar, i de fàcil utilització per part de l'analista. Aquesta opció adquereix major pes si tenim en compte que OOHDM i WebML, a pesar d'emprar notacions pròpies, ni són més expressives ni a priori més senzilles d'entendre i utilitzar. Açò fa que siga necessari el suport per mig d'eines específiques d'aquestes notacions.

OOWS persegueix emprar els conceptes acceptats en els diferents mètodes presentats, fent també especial menció al *Model Navegacional i de Presentació abstracta*, definint un model de domini, i emprant una

notació UML (o pareguda a UML). També, a nivell de requeriments, es proposa començar a partir d'una especificació extesa amb requeriments Web, com es pot veure en [133, 131]. Per a la definició de la funcionalitat i de l'estructura d'informació, s'emprarà OO-Method: 1) per les seues característiques de definició d'un entorn complet de generació de codi, tenint en compte també els aspectes funcionals, i 2) per tenir una eina industrial, *OlivaNOVA*, que materialitza d'una manera concreta els principis d'OO-Method en entorns reals de producció de programari.

OOWS empra la Web semàntica, no per modelar conceptualment l'aplicació Web, sinó per produir continguts anotats semànticament [91].

Capítol 3

OOWS: Un Mètode per Desenvolupar Aplicacions Web

OOWS, acrònim de "*Object-Oriented Web Solution*", és un mètode que aplica els principis de l'Enginyeria Web per proporcionar suport metodològic al desenvolupament d'aplicacions web. OOWS defineix unes extensions sobre el mètode de producció de programari orientat a objectes anomenat OO-Method.

Aquest capítol presenta breument el procés de desenvolupament complet de l'aproximació OO-Method/OOWS, de manera que servisca com a referència per a la resta de treball. Aquest procés està dividit en dues grans etapes en les que d'una banda es crea una especificació del sistema d'informació mitjançant models conceptuals i en una segona etapa es proposa una estratègia d'obtenció d'una implementació automàtica a partir d'aquestos models.

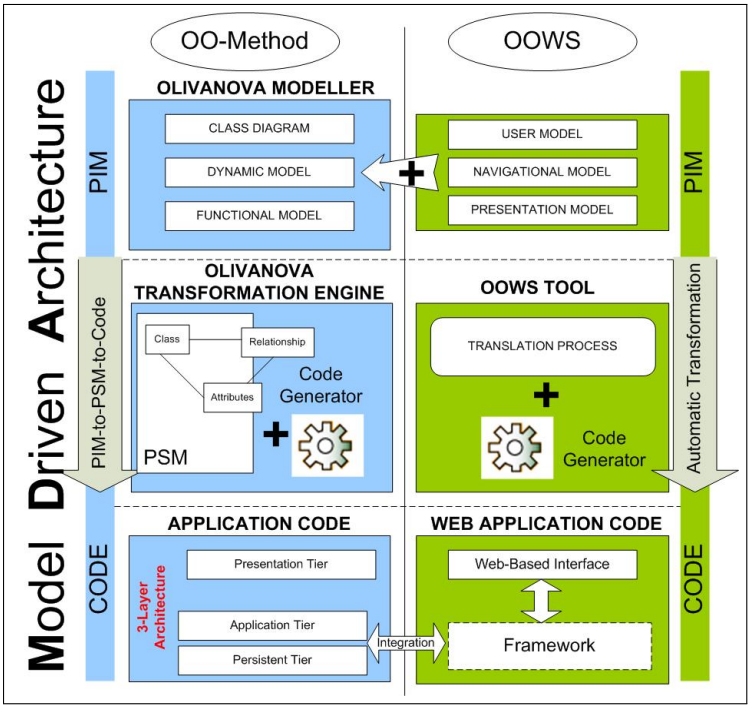


Figura 3.1 – El procés de desenvolupament basat en MDA de OO-Method extés amb OOWS

3.1 Desenvolupament de Programari amb OO-Method

OO-Method [94, 89, 95] (veure part esquerre de la Figura 3.1 és un mètode de producció de programari orientat a objectes que proporciona facilitats de generació automàtica de codi dirigides per models, integrant tècniques d'especificació formal amb notacions orientades a objectes convencionals.

OO-Method proposa un una estratègia de desenvolupament de software basada en dues etapes: l'etapa d'**Especificació del Sistema**, on es construeix una espeficiació d'alt nivell dels requeriments funcionals

dels sistema, i l'etapa de **Desenvolupament de la Solució**, on es segueix una estratègia orientada a la generació de components de software que constitueixen la solució (producte final).

3.1.1 Etapa d'Especificació del Sistema

En l'etapa d'*Especificació del Sistema* es realitzen la següent seqüència d'activitats:

1. **El.licitació de Requeriments Funcionals**. Mitjançant l'ús de tècniques basades en casos d'ús i escenaris es construeix un *esquema conceptual*. En [59] hi ha un extens treball realitzat en aquest àmbit.

2. **Modelatge Conceptual**. Mitjançant l'ús de diferents models es permet la captura i representació del sistema software des de 3 punts de vista diferents:

   - un **Model Estructural** que defineix, mitjançant un paradigma OO, l'estructura d'informació del sistema en base a les seues classes (operacions, atributs, restriccions, etc.) i les relacions entre aquestes (jerarquies d'herència i relacions d'associació, agregació i composició). Tot açò per mig d'un *Diagrama de Classes*
   - un **Model Dinàmic** que descriu les diferents seqüències de vida vàlides dels objectes que pertanyen a cada classe del sistema, a més de les interaccions d'objectes (comunicació entre objectes). Aquestes descripcions es realitzen mitjançant *Diagrames d'Estats* i *Diagrames de Seqüència*, respectivament
   - un **Model Funcional** que captura la semàntica de canvis d'estats per definir els efectes dels serveis[1] i les operacions

[1] En la terminologia OO-Method, "servei" és sinònim d'operació, event, mètode ...

emprant una especificacó formal textual

- un **Model de Presentació** que permet definir unitats de presentació per dotar a l'aplicació d'una interfície d'usuari

Des d'un punt de vista de la MDA [86], OO-Method proporciona un PIM on es recullen els aspectes estàtics i dinàmics del sistemes de programari emprant un conjunt de vistes complementàries, que són definides per aquestos següents models.

3.1.2 Etapa de Desenvolupament de la Solució

Una vegada construit l'Esquema Conceptual d'un sistema d'informació, en l'etapa de *Desenvolupament de la solució* s'aplica un procés sistemàtic per obtenir la solució (completa) final.

Per tal d'aconseguir aquesta solució, es defineix un procés basat en dues etapes: una primera en la que es proposa un **estil arquitectònic** del sistema, i una segona en la que s'apliquen un conjunt de **correspondències** entre abstraccions conceptuals i elements software que composen la implementació de cada capa de l'arquitectura.

OlivaNOVA proporciona un marc de treball operacional, conforme amb MDA on un *Compilador de Models* [25] transforma el PIM (Model Conceptual) en el seu corresponent Producte de Programari. D'una banda, el *Modelador d'OlivaNOVA* [24] ens permet definir gràficament les diferents vistes que descriuen un sistema (l'estructural, dinàmica i funcional). D'altra banda, un conjunt de *Motors de Transformació* compilen aquestes vistes per traduir les primitives de modelatge conceptual definides al PIM en termes de llenguatges d'implementació. D'acord a terminologia emprada en el món MDA, *OlivaNOVA* realitza doncs transformacions "*PIM-to-Code*" (de models a codi).

La generació de les aplicacions es realitza emprant un estil arquitectònic multi-capa, amb una divisió principal en 3 capes:

- **Capa d'Interfície d'Usuari ó de Presentació:** Proporciona els components per desenvolupar la interfície gràfica per interactuar amb l'usuari.
- **Capa de Aplicació:** D'una banda implementa la estructura i funcionalitat de les classes de l'esquema conceptual i d'altra banda proporciona una interfície d'operació per facilitar l'ús per part de la capa de presentació de la funcionalitat implementada.
- **Capa de Persistència:** Es responsabilitza de donar emmagatzemament a les dades (mitjançant sistemes gestors de bases de dades) i dóna una interfície d'accés a dades emmagatzemades amb l'objectiu amagant els detalls dels repositoris de dades a les capes superiors.

Com a resultat de modelar els sistemes amb *OlivaNOVA* i aplicar-los els Compilador de Models, s'obtenen prototipus del sistema que serveixen per validar-los amb els usuaris i realimentar la fase de modelatge amb les propostes de canvi i/o millora. Seguint aquest procés de desenvolupament, s'aconsegueix que les especificacions dels sistemes d'informació siguen documents vius que representen en tot moment el sistema d'informació que s'està tractant de construir.

3.2 Extensió Web

Segons el treball realitzat en [81], les aplicacions web tenen requeriments que no són tractats adientment pels mètodes de producció de programari durant el procés de desenvolupament. Aquestes noves propietats es refereixen a aspectes com ara navegació, presentació i altres característiques avançades com la personalització, la inclusió d'estructures de navegació, definició d'un espai hipermedia, etc. En aquest sentit, OO-Method no

és una excepció i requereix algunes extensions per tractar amb aquestos requeriments.

Agafant OO-Method (*OlivaNOVA*) com a punt de partida, OOWS extén el procés de desenvolupament introduint nous models a la fase de modelatge conceptual per capturar les propietats navegacionals dels sistemes d'informació i nous compiladors de models per a transformar aquestos nous models en els seus corresponents components de programari.

3.2.1 Etapa d'Especificació del Sistema

Seguint la mateixa filosofia que OO-Method, OOWS també implica les dos etapes bàsiques d'una aproximació MDA: *Especificació del Sistema* i *Desenvolupament de la Solució*

En la fase d'*Especificació del Sistema* s'empraran les extensions introduïdes pel mètode per descriure: (1) la societat d'usuaris que tindran accés al sistema, (2) l'estructura navegacional per a cada usuari, mitjançant el seu mapa navegacional, (3) la descripció detallada de cada vista navegacional mitjançant els contextes navegacionals, i finalment (4) la descripció de les propietats de presentació.

Per poder fer la descripció completa de tot açò, OOWS introdueix 3 nous models/diagrames que són explicats amb més detall en posteriors capítols: el Diagrama d'Usuaris, el Model Navegacional i el Model de Presentació.

Un **Diagrama d'Usuaris** (Capítol 5). Permet la identificació i categorització dels usuaris. Determina els tipus d'usuaris que podran interactuar amb el sistema. Aquestos usuaris són organitzats en una estructura jeràrquica, representant inter-relacions entre aquestos. A més a més, aquest model caracteritza als usuaris en tres tipus: anònims, registrats i usuaris "abstractes" o sense permís, descrivint així l'accessibilitat que tindran al sistema.

Un **Model Navegacional** (Capítol 4). Permet la descripció de l'accessibilitat dels usuaris als sistemes en dues fases: la fase de descripció global ("*Authoring-in-the-large*") i la fase de descripció detallada ("*Authoring-in-the-small*"). Proporciona dues primitives bàsiques: el Mapa Navegacional, que estructura l'accessibilitat dels usuaris al sistema, descrivint una vista global d'accés; i el Contexte Navegacional, que representa un punt d'interacció amb l'usuari on és descriu una vista detallada sobre el sistema.

Un **Model de Presentació** (Capítol 7). Permet la descripció de requeriments de presentació mitjançant una sèrie de patrons bàsics. Així, es poden: definir requeriments d'organització i disposició de les dades a visualitzar mitjançant el *patró de disposició d'informació*; establir *criteris d'ordenació* aplicat a les dades (ordenant d'alguna manera la informació a visualitzar); estructurar l'accés a col·leccions de dades mitjançant el *patró de paginació*, creant blocs de dades que es poden explorar, etc.

3.2.2 Etapa de Desenvolupament de la Solució

Aquestos tres nous models introduïts permeten la descripció de la part més orientada a la web. En la fase de *Desenvolupament de la Solució* es descriu com arribar automàticament a la solució final a partir d'aquestos (Capítol 8). El procés de generació de codi implementat pels motors de transformació d'*OlivaNOVA* deu ser extés per tal d'obtenir el codi resultant dels models OOWS. Tanmateix, aquesta extensió ha de ser realitzada d'una manera conservativa amb respecte als transformadors comercials per tal d'assegurar la compatibilitat amb programari ja desenvolupat.

Per tal d'aconseguir aquesta extensió conservativa, s'ha definit un procés de traducció en paral·lel per a la descripció web. Per tant, la integració de ambdós processos (OO-Method i OOWS) es realitza a nivell

d'implementació.

Per dur a terme aquesta tasca, s'ha construït una eina que realitza les transformacions (Capítol 9). Aquesta obté les interfícies basades en Web a partir dels models conceptuals OOWS, ja que aquestos contenen tota la informació necessaria per poder generar el codi. D'acord a MDA, realitzem una transformació automàtica de model-a-codi.

La interfície Web constitueix la capa de presentació (Figura 3.1 a la pàgina 68) de les aplicacions web. Per obtenir les aplicacions web completes, aquesta capa deu ser integrada amb la capa d'aplicació i persitència generades per *OlivaNOVA* (les capes que implementen la funcionalitat del sistema, des d'un punt de vista estàtic i dinàmic). Per realitzar aquesta integració, s'ha desenvolupat un framework (Annexe A) que realitza dues tasques principals: (1) facilita la codificació de l'aplicació, ja que té constructors de més alt nivell que els propis dels llenguatges de programació web i (2) realitza la conexió amb la capa de lògica d'aplicació d'*OlivaNOVA* (a més a més d'altres possibilitats).

3.3 Conclusions

Aquest capítol ha presentat el marc per realitzar una extensió conceptual a un mètode de desenvolupament d'aplicacions de programari, aplicant-se a OO-Method. Aquesta decissió es pren basada en el fet que aquest mètode: disposa d'una forta base formal en la definició dels seus conceptes; aplica els principis del desenvolupament dirigit per models; i gràcies a la seua vessant industrial, hi ha implementacions dels editors de models i generadors de codi.

L'extensió proposada OOWS, ha sigut conservativa amb respecte al procés de desenvolupament que defineix OO-Method: en primer lloc, s'han de realitzar una sèrie d'extensions conceptuals per introduir aquells aspectes rellevants i que actualment manquen als models que guien el

desenvolupament de les aplicacions web; en segon lloc, s'han de definir els patrons o regles de transformació que generaran l'aplicació final.

Capítol 4

Model Navegacional

4.1 Introducció

El Model Navegacional és un dels punts clau i diferenciador entre els mètodes de desenvolupament d'aplicacions de programari per al web. Bàsicament, tots construeixen alguna mena de graf per descriure l'estructura del sistema i d'alguna manera s'associa als nodes d'aquest graf vistes sobre l'estructura d'informació del sistema (en algunes aproximacions, inclús la descripció navegacional duu a la creació de l'estructura d'informació).

En aquest capítol es presenta el Model Navegacional introduït pel mètode OOWS. Abans que res, es fa una reflexió sobre el concepte de Navegació. Després es presenta la fase de Modelatge Navegacional, que intenta capturar els requeriments navegacionals dels sistemes web. Aquest modelatge proporciona mecanismes i primitives per representades clarament les dues fases d'autoria del sistemes hipermedials ("*Authoring-in-the-large*" i "*Authoring-in-the-small*"). Una a una s'analitzen les primitives navegacionals i es mostraran exemples amb elles.

4.2 El Concepte de Navegació

Abans de començar la definició del model navegacional, cal deixar clar quin és el concepte principal d'aquest: la navegació. Cal comentar que als principals llocs que tracten de donar suport a la comunitat que engloba l'Enginyeria Web s'ha intentat arribar a un consens en quan a la seua definició, però encara no s'ha proposat una definició clara i amplament acceptada.

Encara que pareix obvi, a causa de l'experiència de la gent emprant la web, molts autors no distingueixen la navegació de altres funcionalitats de l'aplicació, i altres vegades la inclouen com qualsevol event que provoca canvis en la interície. Queda clar, això sí, que la navegació és una característica emergent de les aplicacions web. Encara que alguns autors no la consideren allò suficientment important per donar-li l'atenció necessària durant l'anàlisi del sistema [111].

D'altra banda, la majoria dels mètodes de desenvolupament d'aplicacions Web tracten la navegació com un punt determinant d'aquestes aplicacions i proporcionen models i notació per especificar-la. Bàsicament proporcionen mecanismes per descriure "continguts" (nodes) que són enllaçats entre sí ("*links*") definint el concepte de navegació.

Amb aquestes restriccions sobre el concepte de navegació i la seua implicació a nivell conceptual, queden més clars quins són els requeriments navegacionals front a d'altres tipus (per exemple, els requeriments funcionals). La descripció navegacional d'un sistema consistirà en la descripció d'uns requeriments no-funcionals que permetrà millorar la interacció de l'usuari amb aquestos sistemes.

4.3 Modelatge Navegacional

Si tornem al Procés de Desenvolupament de programari proposat per aquesta tesi, com a unió OO-Mehtod/OOWS, arribem a la fase de

Modelatge Navegacional després d'haver fet una descripció (almenys inicial) de l'estructura d'informació del sistema (Diagrama de Classes). És a dir, en aquest punt ja tenim determinats els objectes (i les seues classes) que són rellevants en el sistema, i el que es pretèn és aconseguir una descripció navegacional de continguts.

El primer que proposa OOWS és fer una descripció general de la navegabilitat del sistema mitjançant la descripció d'un mapa navegacional per a cada tipus d'usuari (veure Capítol 5). Aquesta fase sol anomenar-se en la comunitat Web com "*Authoring-in-the-large*". Una vegada definida aquesta descripció general, es farà la descripció detallada del contingut dels contextes navegacionals abans detectats. Aquesta fase rep el nom de "*Authoring-in-the-small*".

Les següents seccions presenten detalladament cadascuna de les primitives que composen el model navegacional, tant a nivell de descripció general com detallada.

Amb respecte al "*Authoring-in-the-large*", el model proporciona les primitives: *mapa navegacional*, *enllaç navegacional* i *node navegacional*.

La resta de primitives descrites a aquest capítol es refereixen a l'especificació detallada de les propietats navegacionals que es fa a la fase de "*Authoring-in-the-small*".

4.4 Mapa Navegacional, "Authoring-in-the-large"

Un **mapa navegacional** recull les propietats navegacionals d'un usuari concret del sistema. Aquest mapa descriu una *vista global del sistema* o "*Authoring-in-the-large*", creant una estructura d'accessibilitat a continguts d'informació i a execució d'operacions.

És doncs l'eina que permetrà descriure la manera en que un tipus d'usuari concret podrà navegar pel sistema per tal de consultar informació i dur a terme certa funcionalitat, atenent a les seues responsabilitats

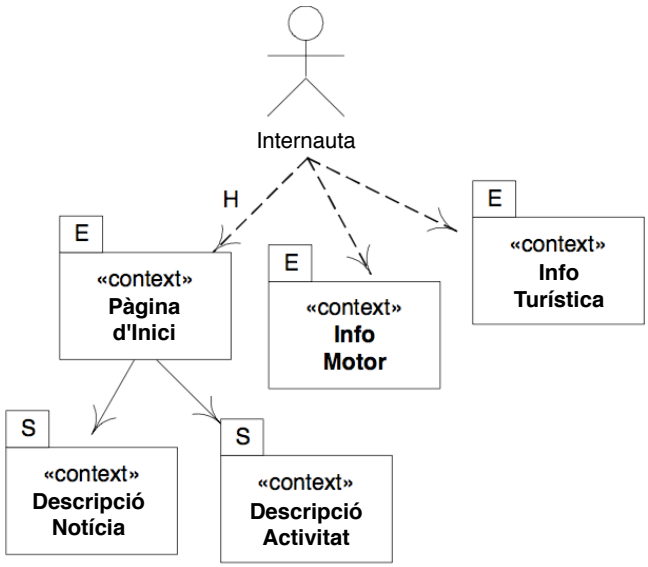


Figura 4.1 – Exemple de Mapa Navegacional

i privilegis com a usuari dins el sistema.

Un mapa navegacional (Figura 4.1) es representa mitjançant un graf dirigit, format per *nodes* (anomenats nodes navegacionals) i *arcs* (anomenants enllaços navegacionals).

Els **nodes** dels mapa navegacional representen punts d'interacció del sistema amb l'usuari, on se li proposen uns continguts per poder realitzar una certa tasca. Aquesta tasca pot estar relacionada amb la consulta d'informació, amb l'activació i execució d'una operació o amb el proveïment d'enllaç que habiliten navegació a altres nodes.

Els **enllaços** navegacionals defineixen els nodes navegacionals que són accessibles des d'un node navegacional determinat.

D'aquesta manera, i sense haver d'especificar els detalls de contingut de cada node i assumint que cada node té un objectiu dins el sistema

(relacionat amb la tasca o tasques que permet realitzar), és pot construir una estructura d'accessibilitat al sistema que permet organitzar les tasques i responsabilitats d'un usuari.

La majoria d'aproximacions metodològiques també introdueixen a nivell de mapa estructures d'accés (normalment índexos) per tractar de millorar la navegabilitat pel sistema . Tanmateix, el mètode OOWS no ho introdueix d'aquesta manera just per tractar de no explosionar encara més els mapes navegacionals. Una discussió detallada de perquè no aparèixen aquestes estructures d'accessibilitat es pot trobar en la Secció 6.2 del Capítol 6.

Una propietat que s'expresa a nivell de mapa navegacional és la de **node per defecte** (o “*Home*”). Aquest és el node al que l'usuari obtindrà només accedisca al sistema. Sols podrà ser un node per defecte algun dels nodes d'exploració (veure Secció 4.4.1), que són del tipus “sempre accessibles”. En la Figura 4.1 a la pàgina anterior pot veure's que està marcat com “Home” el node anomenat *Página de Inicio*.

4.4.1 Nodes Navegacionals

Un node navegacional representa un punt d'interacció de l'usuari amb el sistema i representa un contingut o vista parcial sobre el sistema d'informació.

Hi ha dos tipus de nodes navegacionals: contextes i subsistemes. Els *contextes* defineixen un contingut d'informació i funcionalitat accessible a l'usuari (es veuran a la Secció 4.5). Els *subsistemes* permeten organitzar una estructura d'accessibilitat a un conjunt de nodes navegacionals (es veuran a la Secció 4.5.5)

Existeixen dos tipus de nodes atenent a la seua alcançabilitat dins el mapa navegacional:

- **Nodes d'Exploració.** Aquest tipus de nodes són sempre alcançables des de qualsevol altre node del mapa. Açò implica que haurà

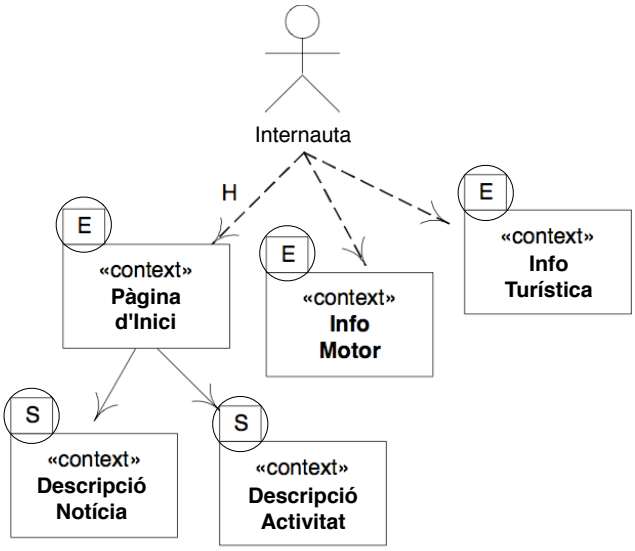


Figura 4.2 – Nodes d'Exploració i de Seqüència

d'existir un enllaç navegacional (de tipus exploració) des de cada node del sistema a tot node d'exploració. Un d'aquestos pot estar marcat com *node per defecte*. Gràficament s'etiqueten amb una E.

- **Nodes de Seqüència.** Aquest tipus de nodes sols seran alcançables seguint un camí navegacional [1] predefinit. És a dir, per poder alcançar-los, s'haurà de navegar primer per altres nodes que li donen accés. Gràficament s'etiqueten amb una S.

La Figura 4.2 mostra un exemple de mapa navegacional on es poden veure els nodes d'exploració (*Página de Inicio*, *Info Motor* i *Info Turistica*) i els de seqüència (*Descripción Noticia* i *Descripción Noticia*).

[1] Seqüència enllaçada de nodes

4.4.2 Enllaços Navegacionals

Un enllaç navegacional defineix una relació d'alcançabilitat direccional entre dos nodes de navegació. És a dir, si entre els nodes navegacionals N1 i N2 existeix un únic enllaç navegacional, amb la direcció de N1 cap a N2, implica que el node N2 serà accessible des del node N1, però no a l'inrevés.

Hi ha dos tipus d'enllaços navegacionals atenent al tipus d'alcançabilitat que es vulga definir: els enllaços d'exploració i els de seqüència:[2]

Els **enllaços d'exploració** permeten definir una relació d'alcançabilitat als nodes que es consideren inicis d'activitats o tasques de l'usuari. Aquestos nodes doncs, serveixen per definir els punts d'entrada al sistema. Algunes aproximacions solen anomenar a aquestos enllaços, *enllaços estructurals*.

Els enllaços d'exploració estan directament relacionats amb els nodes d'exploració: un. Gràficament es representen mitjançant una fletxa amb línia discontinua que sorgeix del tipus d'usuari sobre el que es defineix el mapa i acaba en el node d'exploració. Aquesta representació tracta d'aconseguir dues coses: (1) indicar que l'usuari té directament accés a tots aquestos nodes (ja que són d'exploració) i (2) compactar la representació gràfica del mapa, ja que el graf “real” hauria de tenir per a cada node del graf un enllaç cap a tots els altres nodes d'exploració.

Els **enllaços de seqüència** permeten definir una relació d'alcançabilitat entre nodes en base a una relació entre els continguts d'informació que hi proporciona cada node. És a dir, un enllaç de seqüència està associat en el node orige a una informació (que forma part de la vista que defineix el node) i enllaça amb el node destí canalitzant o duent aquesta informació com a “contexte” [3] de la navegació. Permet definir relacions

[2] Aquest concepte està íntimament relacionat amb el concepte de nodes d'exploració i nodes de seqüència

[3] Contexte en aquest cas significa propietat o àmbit

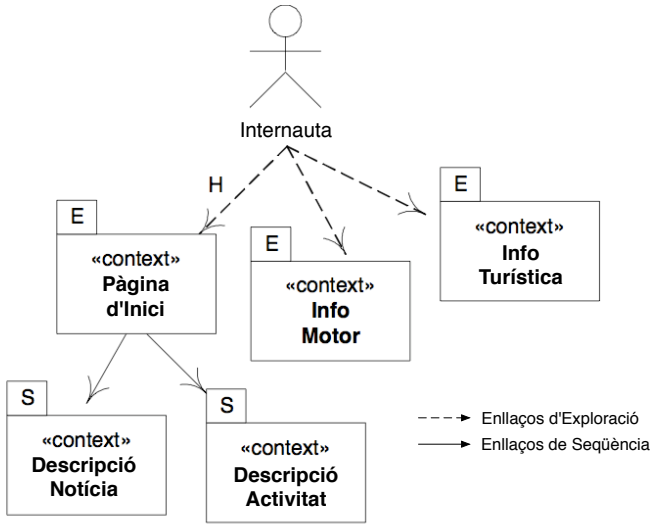


Figura 4.3 – Enllaços d'exploració i seqüència

de navegació mitjançant informació dins l'estructura navegacional del sistema.

Els enllaços de seqüència estan directament relacionats amb les relacions navegacionals (Secció 4.5.3). Gràficament es representen mitjançant una fletxa amb línia continua.

Si observem la Figura 4.3, podrem veure que hi ha 2 enllaços d'exploració (fletxes discontínues) i 3 enllaços de seqüència (fletxes contínues). Com es pot observar, tots els enllaços de seqüència tenent com a orige i com a destí un node navegacional. D'altra banda, els enllaços d'exploració tenen com a orige l'*usuari propietari* del mapa navegacional i com a destí un node navegacional.

4.5 Contextes Navegacionals, "Authoring-in-the-small"

Així com el mapa navegacional (amb els nodes i enllaços) és la base per definir la vista global del sistema, els contextes navegacionals són la base per definir el "*Authoring-in-the-small*".

Un contexte navegacional és un tipus de node navegacional que defineixen una ***vista parcial del sistema*** *sobre informació i funcionalitat del sistema, amb l'objectiu de permetre-li a l'usuari dur a terme una certa activitat o tasca.* Aquesta vista es defineix com una projecció sobre el Diagrama de Clases definint un conjunt de dades (atributs de classes) i funcionalitat (operacions de classes) que seran accessibles a l'usuari quan navegue a aquest contexte. S'anomena **població del contexte** al conjunt de dades que el contexte ha de recuperar i fa visible a l'usuari.

Gràficament, un contexte navegacional es representa per un paquet UML estereotipat amb la paraula reservada ≪contexte≫.

Atenent al tipus d'alcançabilitat del contexte navegacional, n'existeixen de dos tipus: contextes de seqüència i contextes d'exploració.

Un **contexte de seqüència** és aquell que sols és accessible seguint algun camí navegacional predefinit. Açò és així perquè interessa que a l'arribar a aquest tipus de contextes, l'usuari haja seleccionat alguna informació en el node previ i en aquest contexte es mostre informació detallada d'aquesta informació.

La Figura 4.4 a la pàgina següent mostra mapa navegacional amb un contexte de seqüència, anomenat *Guests* ("Convidats"). Com es pot observar, no es pot arribar a ell directament (no té cap enllaç d'exploració definit a partir de l'usuari membre) i només es pot arribar a ell seguint algun dels camins navegacionals següents: **-Projects-Guests* ó **-Members-Guests*[4]. Suposem que estem al contexte *Projects*. Tenint

[4]Aquesta notació intuitiva representa dos camins navegacionals en els que, siga

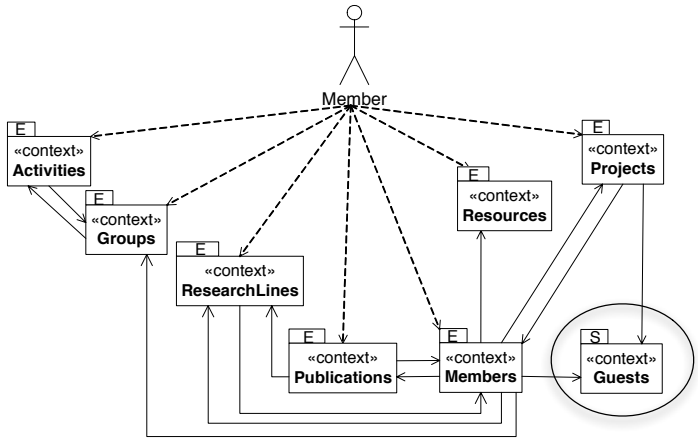


Figura 4.4 – Mapa Navegacional: Contextes de Seqüència

en compte que al diagrama de classes (estructura d'informació) hi ha una relació entre la classe *Project* i la classe *Guest* que indica que hi ha convidats que participen en un projecte.

En aquest contexte *Projects*, tal com s'especifica al mapa navegacional, existiran enllaços que duguen al contexte *Guest* per seqüència, és a dir, seleccionant i duent alguna informació entre els contextes. Allò més normal seria que que al contexte *Projects* es vegeren els noms dels convidats que hi participen, i que al seleccionar un d'aquestos convidats, es veurà informació detallada d'aquest en el contexte *Guests*. De la mateixa manera, la classe *Member* té una relació d'associacio amb la classe *Guest*, indicant quins convidats estan a càrrec de quin member. Obtenir aquestos convidats a partir del contexte *Members* per seleccionar un dels membres concrets, podria ser un escenari que també faria viable aquest mapa navegacional.

quin siga l'inici, al final s'accedeix primer al node *Projects* o *Members* abans d'accedir al node *Guests*

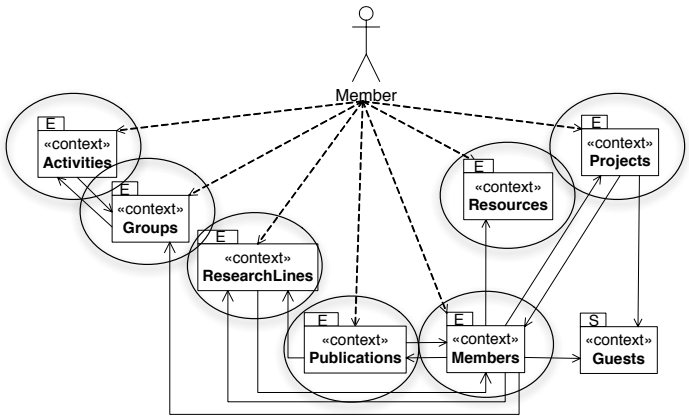


Figura 4.5 – Mapa Navegacional: Contextes d'Exploració

Un **contexte d'exploració** té la particularitat que defineix un contingut sempre accessible per a l'usuari. Açò vol dir que en qualsevol moment, l'usuari pot demanar-li a l'aplicació que el duga a aquest contexte. Com s'ha vist abans en la Secció 4.4.2, tot contexte d'exploració té associat al mapa navegacional un enllaç d'exploració que l'uneix amb l'usuari, indicant així que és sempre accessible. Cada vegada que l'usuari empre aquest enllaç d'exploració per arribar al contexte, el contexte li proporcionarà la població completa (totes aquelles instàncies que satisfaguen la vista del contexte).

La Figura 4.5 mostra un mapa navegacional amb 7 contextes d'exploració: *Activities*, *Group*, *Publications*, *ResearchLines*, *Members*, *Resources* i *Projects*. Tots aquestos són directament accesibles. Suposem que el al contexte *Publications* mostra totes les publicacions que hi ha disponibles al sistema. Si accedim a ell per mig de l'enllaç d'exploració, el sistema ens mostrarà la població completa de publicacions que té disponibles.

Però, a més a més, un contexte d'exploració pot comportar-se com un

contexte de seqüència si definim un enllaç de seqüència des de qualsevol altre node fins el contexte. Amb aquest requeriment, el que estem representant és que si accedim a un contexte d'exploració creuant un enllaç de seqüència, enlloc de mostrar tota la població del contexte mostrarà sols aquella que estiga relacionada amb la informació que hajam seleccionat en el node orige (enllaç de seqüència).

Així, si revisem la Figura 4.5 a la pàgina anterior podrem adonarnos que també podem accedir al contexte *Publicacions* per l'enllaç de seqüència a partir del contexte *Members* (tras seleccionar un membre), sols es mostraran les publicacions d'aquest membre seleccionat i no les de tot el sistema

4.5.1 Unitats d'Abstracció d'Informació, UAI

Un Contexte Navegacional defineix un contingut que és accessible a un determinat usuari quan hi accedeix al contexte. A causa del tipus de necessitats actuals en els sistemes web, és molt comú que aquest contingut siga en realitat un agregat de continguts, on es mostre diferent tipus d'informació.

Una **Unitat d'Abstracció d'Informació ó UAI** defineix un contingut concret i completament relacionat entre sí sobre el Diagrama de Classes. D'altra banda, un contexte es defineix com un agregador de UAIs. D'aquesta manera, un contexte es converteix en un agregador de continguts, possiblement no relacionats[5]. Per a una discussió més detallada sobre l'agregació de continguts, veure la Secció 6.3 del Capítol 6.

En el cas més simple, un contexte podrà contenir una única UAI. En aquestos casos, el model permet no especificar la UAI explícitament (encara que hi existeix implícitament). A més a més, aquesta UAI ha de ser obligatòriament de tipus contextual.

[5]Tot el contingut d'una UAI està relacionat entre sí, però no ho està amb respecte d'altres UAIs

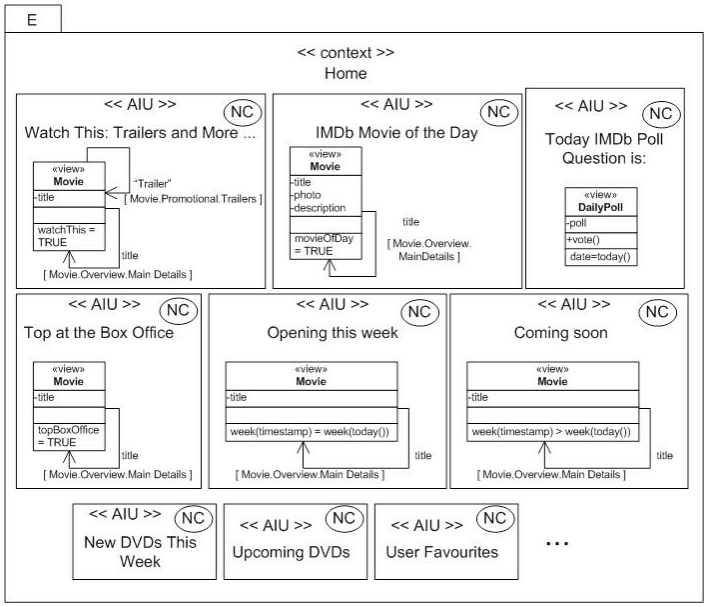


Figura 4.6 – Unitats d'Abstracció d'Informació d'un contexte

Les UAIs estan formades per classes navegacionals i relacions navegacionals, les quals li permeten definir l'estructura d'informació i funcionalitat que definirà la vista. La Figura 4.6 mostra un exemple de contexte amb UAIs: .

4.5.2 Classes Navegacionals

Una **Classe Navegacional** és una *vista* sobre una classe del Diagrama de Classes. Aquesta vista defineix una projecció sobre les propietats de la classe que formaran part de la vista. Aquestes propietats es refereixen a atributs i operacions de la classe.

Una classe navegacional es representa gràficament com una classe en

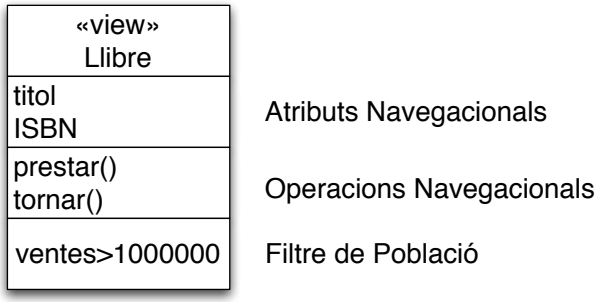


Figura 4.7 – Classe Navegacional

UML estereotipada amb ≪vista≫[6]. Aquesta representació gràfica de la classe, enlloc de tenir els dos calaixos típics d'UML per a representar atributs i operacions, en té tres calaixos: el primer, per a indicar els **atributs navegacionals** visibles; el segon, per a indicar les **operacions navegacionals** que seran activables en aquest contexte; i el tercer per indicar la **condició de filtrat** que hi pot existir. Per simplificar la seua representació, és comú que no aparega aquell calaix on no hi haja res especificat.

La Figura 4.7 mostra una classe navegacional.

Un **atribut navegacional** és un atribut que pertany a la classe (en el Diagrama de Classes) i sobre el que es demana "visibilitat". Dit d'altra manera, per a que una propietat (atribut) d'una classe siga visible en un node (contexte navegacional), ha d'aparèixer com atribut navegacional en una classe navegacional.

De la mateixa manera que els atributs navegacionals, una **operació navegacional** és una operació que pertany a la classe (en el Diagrama de Classes) que serà accessible des de la vista actual, permetent la seua execució. És a dir, en una UAI, si apareix alguna operació navegacional,

[6]Per raons de llenguatge, de vegades s'expressa aquest estereotip en la seua forma anglesa ≪view≫

està indicant que l'usuari tindrà accés per llançar (executar) l'operació associada.

Finalment, en una classe navegacional es pot definir una que permet restringir la població de la classe que s'ha de recuperar. Sols formaran part de la població de la UAI aquelles instàncies de la classe que complisquen la condició assenyalada. Aquestes condicions de filtrat s'especifiquen mitjançant fòrmules semblants a OCL, en el tercer calaix de la classe navegacional. La Figura 4.7 a la pàgina anterior mostra la condició de filtrat

D'altra banda, hi ha dos tipus de classes navegacionals: la classe directora i les classes complementàries.

Una **classe navegacional directora** és la principal dins una UAI, i *sols hi pot haver una*. Defineix quin és el punt de partida (classe del Diagrama de Classes) sobre la que s'inicia a definir la vista.

Una **classe navegacional complementària** permet completar la vista d'una altra classe navegacional (directora o complementària) sobre la que es defineix. Aquestes classes complementàries sempre van unides a una altra classe navegacional per una relació navegacional (definides a la Secció 4.5.3).

4.5.3 Relacions Navegacionals

Les classes navegacionals dins una UAI estan relacionades per *relacions binàries unidireccionals* anomenades **Relacions Navegacionals**.

L'objectiu d'una relació navegacional és definir un requeriment per recuperar informació complementària (creant les classes complementàries), relacionada per alguna relació estructural (en el Diagrama de Classes) entre les classes. És per açò que es defineix sobre relacions d'associació-compossició-agregació i també sobre relacions d'herència.

El contexte navegacional de la Figura 4.8 a la pàgina següent mostra una relació navegacional (representada com una fletxa entre les classes)

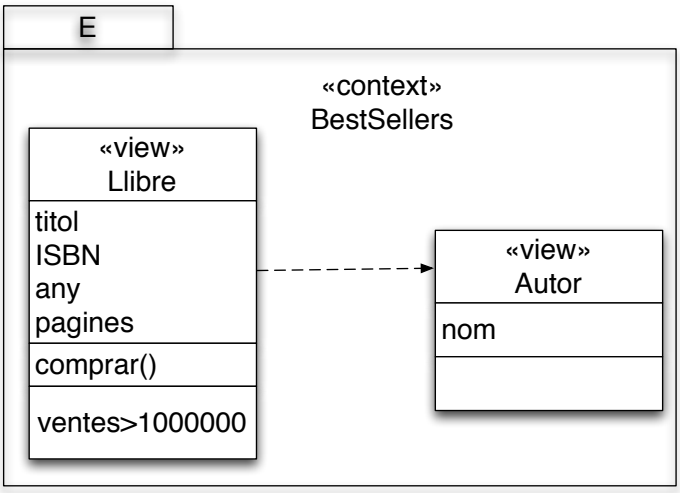


Figura 4.8 – Relacions Navegacionals

que representa una recuperació d'aquells *Autors* que han escrit un *Llibre*.

Per a eliminar qualsevol tipus d'ambigüitat en el cas de múltiples relacions entre dues classes concretes, les relacions navegacionals han d'incloure un *atribut de rol* (representat gràficament com */atribut-de-rol/*) que indicarà el nom de la relació a la que es fa referència.

Així, per exemple, a les Figures 4.9 i 4.11, s'aprecia que s'ha especificat l'atribut de rol /Escrit_per/. Açò pot ser causat perquè entre la classe Llibre i la classe Autor podrien existir almenys dues relacions d'associació: una per indicar els autors que han *escrit* un llibre (relació *Escrit_per*) i una altra relació que indique quins autors han *revisat* un llibre (relació *Revisat_per*). D'aquesta manera es lleva l'ambigüitat i es sap en tot moment sobre quina relació estructural s'està projectant la relació navegacional.

A més a més, atenent a si la Relació Navegacional defineix o no una capacitat de navegació a un node destí, existeixen dos tipus de

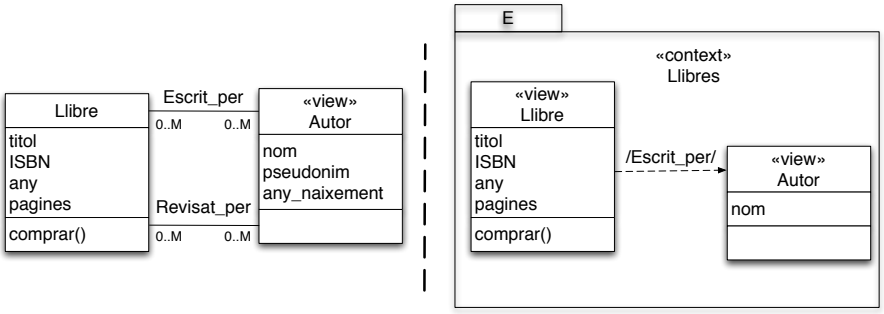


Figura 4.9 – Relació de Dependència de Contexte

relacions navegacionals: la relació de dependència de contexte i la relació de contexte.

4.5.3.1 Relacions de Dependència de Contexte

Una **Relació de Dependència de Contexte** representa un requeriment de recuperació d'informació d'una classe relacionada a partir d'una classe navegacional inici. Cada vegada que es recupere una instància de la classe inicial de la relació de dependència, es recuperaran totes les instàncies de la classe final de la relació existents en el sistema que n'estiguen relacionades.

Es representa gràficament mitjançant una fletxa discontinua, ⇢

La Figura 4.9 mostra un exemple de relació de dependència de contexte, aplicat al contexte *Llibres*. Segons aquesta relació, per a cada llibre que es recupere al contexte (del que es mostrarà el ISBN i el títol) s'haurà de proporcionar informació addicional sobre el nom dels autors que l'han escrit (relació /Escrito_por/ de tipus "molts-a-molts").

La Figura 4.10 a la pàgina següent mostra una possible implementació d'aquest contexte navegacional, on es veu les implicacions d'una relació de dependència de contexte. Com es pot observar, per a cada llibre es proporciona la informació del nom dels autors d'aquest.

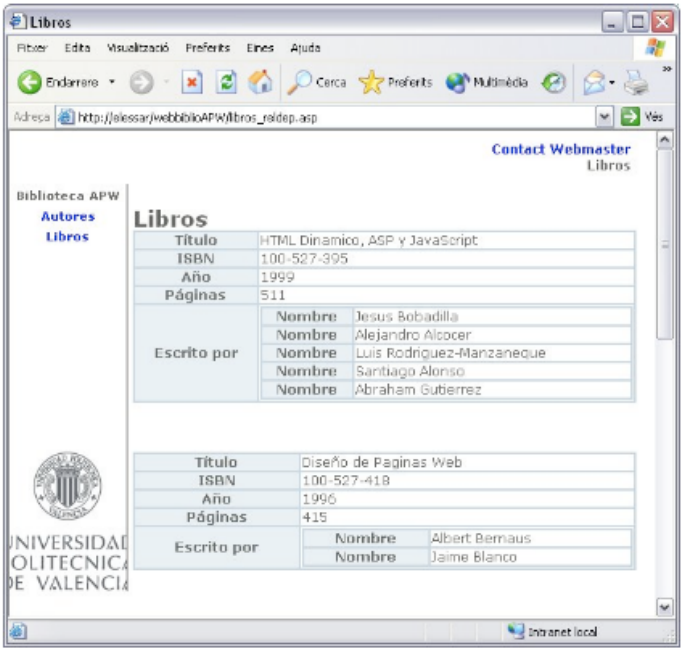


Figura 4.10 – Implementació d'una Relació de Dependència de Contexte

4.5.3.2 Relacions de Contexte

Una **Relació de Contexte** representa la mateixa recuperació d'informació que les relacions de dependència de contexte, però ademés permet definir una capacitat de navegació a un contexte navegacional destí. Aquesta relació crea un enllaç de seqüència en el mapa navegacional (veure Secció 4.4.2). Gràficament es representa mitjançant una fletxa continua, →.

Per tal de completar l'enllaç de navegació que defineix, cal especificar les propietats "atribut de contexte" i "atribut d'enllaç".

Un **[Atribut de contexte]** indica el contexte destí de la navegació. És a dir, un contexte navegacional que existisca al mapa navegacional sobre

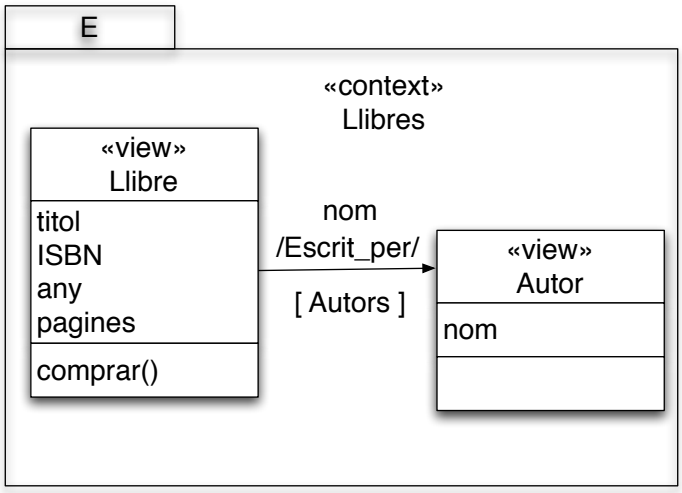


Figura 4.11 – Relació de Contexte

el que es defineix. Gràficament es representa com [contexte-destí].

Un **Atribut d'enllaç** especifica l'atribut que servirà com a “àncora” per activar la navegació al contexte destí. Habitualment, l'atribut d'enllaç pertany a la classe navegacional destí en la relació de contexte. Per qüestions de facilitat de comprensió de cara a l'usuari, de vegades és interessant definir aquesta àncora com una “etiqueta” (un text estàtic). En aquest cas, enlloc d'especificar un atribut navegacional, es ficarà entre dobles-cometes el text que es desitja.

La Figura 4.11 mostra el mateix exemple del contexte *Llibres*, però ara definint una capacitat de navegació emprant el nom de l'autor com enllaç al contexte destí [Autors].

La Figura 4.12 a la pàgina següent mostra una possible implementació d'aquesta relació de contexte. Com es pot apreciar, ara els noms dels autors dels llibres s'han convertit en enllaços que donen accés (per seqüència) al contexte destí [Autors], duent com a informació contextual

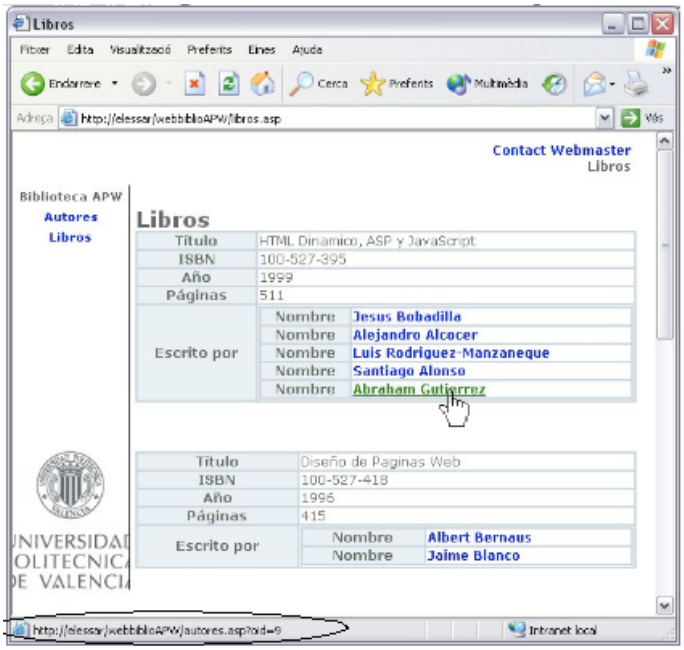


Figura 4.12 – Implementació d'una Relació de Contexte

l'autor seleccionat. El contexte destí, *Autors*, recuperarà només la informació referent a l'autor indicat.

Atenent al tipus d'informació contextual que "envia" una relació de contexte al contexte destí, podem distingir dues relacions: *Relacions de Contexte d'Objecte* i *Relacions de Contexte de Relació*.

4.5.3.3 Relacions de Contexte de tipus Objecte

Una *Relació de Contexte d'Objecte* defineix la informació contextual de la navegació com un objecte. És a dir, fruit de la navegació, se seleccionarà un objecte (instància concreta d'una classe navegacional al contexte orige) i aquesta es "durà" com a informació contextual. Al contexte

destí, la classe directora de les UAIs contextuals hauran de ser del mateix tipus que l'objecte, i quan es produisca la navegació, recuperaran només la informació de l'objecte passat contextualment.

Per temes de qualitat, que tenen a veure amb la coherència i facilitat de comprensió per part de l'usuari final de les aplicacions, es recomana que l'*atribut d'enllaç* (atribut emprat com àncora en la navegació) siga un atribut de la classe destí de la relació de contexte. Açò és deu al fet que l'objecte que seleccionem pertany a la classe navegacional destí de la relació, que a la seua vegada serà la classe principal en el contexte destí. El que s'aconsegueix amb aquesta recomanació és que l'usuari, al seleccionar informació sobre un tipus d'objecte, aconseguisca al final informació sobre aquest tipus d'objecte i no d'un altre tipus (que podria provocar confussió).

Exemple: Suposem que estem explorant un catàleg de llibres, on cada llibre pot ser escrit per varios autors. Es desitja definir una navegabilitat de manera que en un contexte es mostre tota la informació d'un llibre, incloent el nom dels autors, i que, al seleccionar un d'aquestos autors, s'obtinga informació addicional de l'autor seleccionat.

La Figura 4.9 a la pàgina 93 mostra el Diagrama de Classes que s'emprarà per a aquest exemple. Com es pot observar, representa els requeriments d'estructuració d'informació, on un *Llibre* pot estar escrit (mitjançant la relació /Escrit_per/) per més d'un autor.

Per representar el requeriment navegacional descrit és necessari emprar una relació de Contexte d'Objecte, associada al tipus d'objecte *Autor*, i s'emprarà com a àncora el nom de l'autor. La Figura 4.13 a la pàgina següent mostra el Mapa Navegacional associat, i els contextes *Llibres* i *Autors* que representen aquest requeriment.

Finalment, una possible implementació d'aquest escenari podria veure's a la Figura 4.14 a la pàgina 99, on es veu la implementació del contexte *Llibres* on, al passar per damunt d'un enllaç associat al nom de

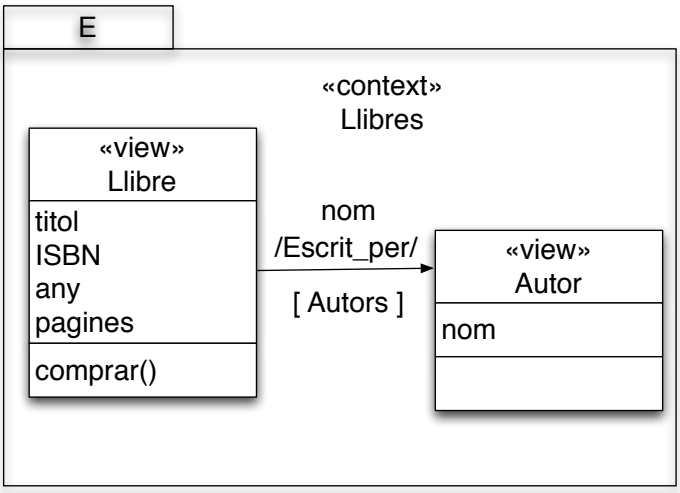


Figura 4.13 – Relació de Contexte d'Objecte

l'autor, duu a la pàgina *autores.asp* (que implementa el contexte *Autors*) passant com informació contextual l'objecte seleccionat (l'identificador d'objecte dels autors).

4.5.3.4 Relacions de Contexte de Relació

Una *Relació de Contexte de Relació* defineix la informació contextual de la navegació com una relació. És a dir, fruit de la navegació, se seleccionarà una relació entre classes associada a un objecte i aquesta es "durà" com a informació contextual (junt a l'objecte). Al contexte destí, la classe directora de les UAIs contextuals hauran de ser alguna classe relacionada (en el diagrama de classes) amb la classe a la que pertany l'objecte per la relació passada. Quan es produisca aquesta navegació, el contexte destí recuperarà només totes les instàncies relacionades amb la relació i l'objecte passats contextualment.

El que aconseguim amb aquesta primitiva és que no es mostre infor-

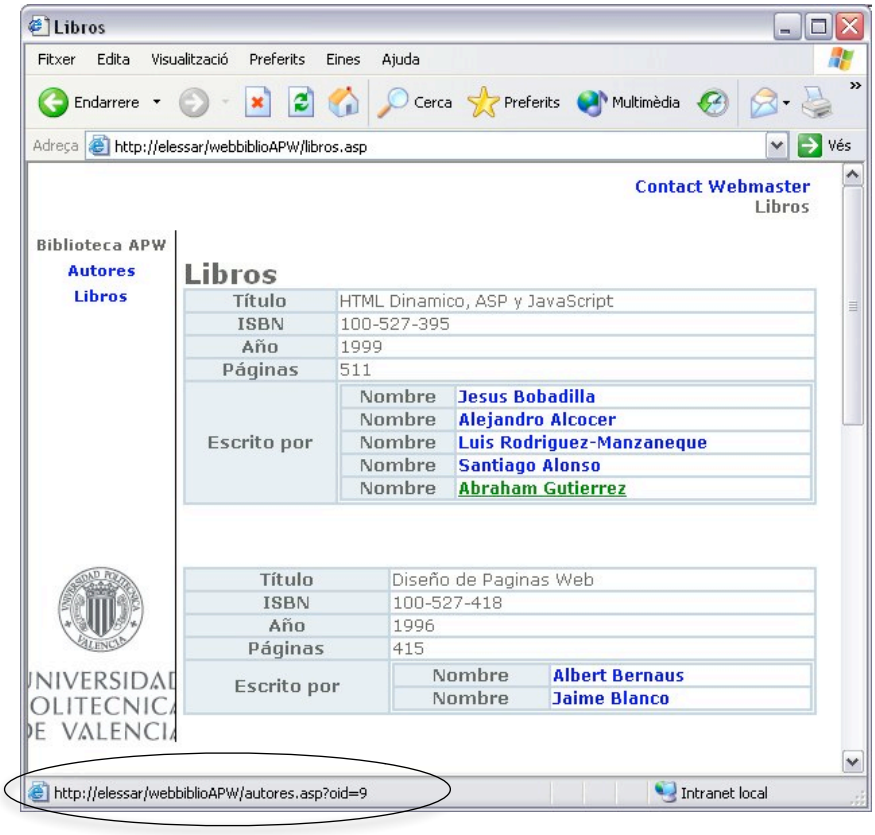


Figura 4.14 – Implementació d’una Relació de Contexte d’Objecte

mació sobre els objectes relacionats, i aquestos es mostren al contexte destí. Aquesta primitiva és adient quan la població de relacionats és suficientment gran, o quan aquesta població té una autonomia semàntica (tenen significat per sí mateix). És per açò que quan s’especifica una relació de contexte de relació, la classe navegacional destí de la relació ha d’estar buida: no ha de tenir atributs navegacionals, per no forçar la recuperació d’instàncies.

Amb respecte a l'*atribut d'enllaç*, hi ha vàries alternatives:

- *No s'especifica atribut d'enllaç.* En aquest cas apareixerà el nom de la relació com el text que servirà d'àncora en la navegació. Aquest cas és el més desitjable, ja que directament relaciona l'àncora que seleccionarà l'usuari amb el contingut d'informació que es recuperarà al destí. En cas necessari, es podria emprar un text estàtic definit per l'usuari en aquest contexte enlloc del nom de la relació, si amb açò es guanya en comprensió.

- *S'especifica un atribut de la classe naveagacional orige* de la relació. En aquest cas, l'àncora s'associa amb un atribut de la classe navegacional orige. Té l'inconvenient de cara a l'usuari final que aquest selecciona un tipus d'objecte i en el destí de la navegació veurà altre tipus d'objectes (aquells relacionats per la relació de contexte). Malgrat açò, en alguns escenaris és adient emprar aquest tipus d'àncora.

Exemple: Suposem que estem explorant un catàleg de llibres (el mateix que abans), on cada llibre pot ser escrit per varios autors. Es desitja definir una navegabilitat de manera que en un contexte es mostre tota la informació d'un llibre, però enlloc de proporcionar informació sobre els autors, es desitja que es mostre un enllaç (de l'estil de "*Autors que han escrit el llibre*") que duga al contexte destí *Autors* on es proporcione informació de tots els autors que han escrit el llibre seleccionat.

La Figura 4.9 a la pàgina 93 mostra el Diagrama de Classes. Un *Llibre* pot estar escrit (mitjançant la relació /Escrit_per/) per més d'un autor.

Per representar el requeriment navegacional descrit és necessari emprar una relació de Contexte de Relació, associada al tipus d'objecte *Llibre* i emprar la relació /Escrit_per/. Com atribut d'enllaç (àncora

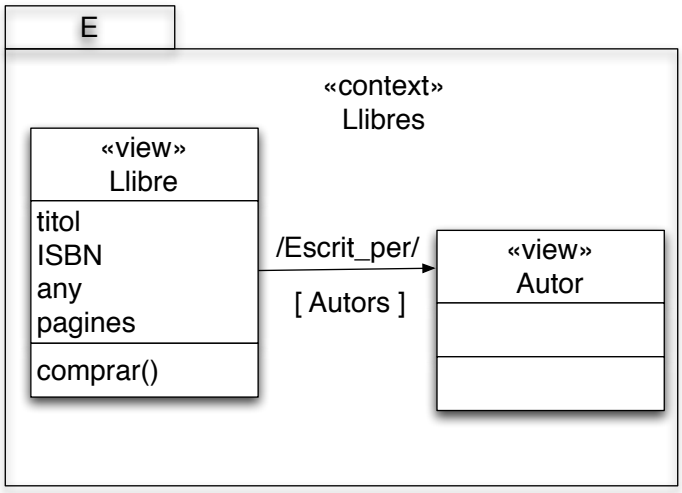


Figura 4.15 – Relació de Contexte de Relació

de la navegació) podem emprar el propi nom de la relació, de manera que no cal que n'especifiquem cap. La Figura 4.15 mostra el Mapa Navegacional associat, i els contextes *Libros* i *Autores* que representen aquest requeriment.

Finalment, una possible implementació d'aquest escenari podria veure's a la Figura 4.16 a la pàgina següent, on es veu la implementació del contexte *Llibres* on, al passar per damunt d'un enllaç associat al nom de la relació *Escrit per* (en aquest cas no es veuen els autors en aquesta pàgina), duu a la pàgina *autores.asp* (que implementa el contexte *Autors*) passant com informació contextual l'objecte seleccionat (l'identificador d'objecte dels autors) i la relació que s'està emprant (*Escrit_per*).

4.5.4 Enllaços de Servei

Un **Enllaç de Servei** va associat a una operació navegacional i defineix una navegació que es produirà després de l'execució d'aquest servei. Per

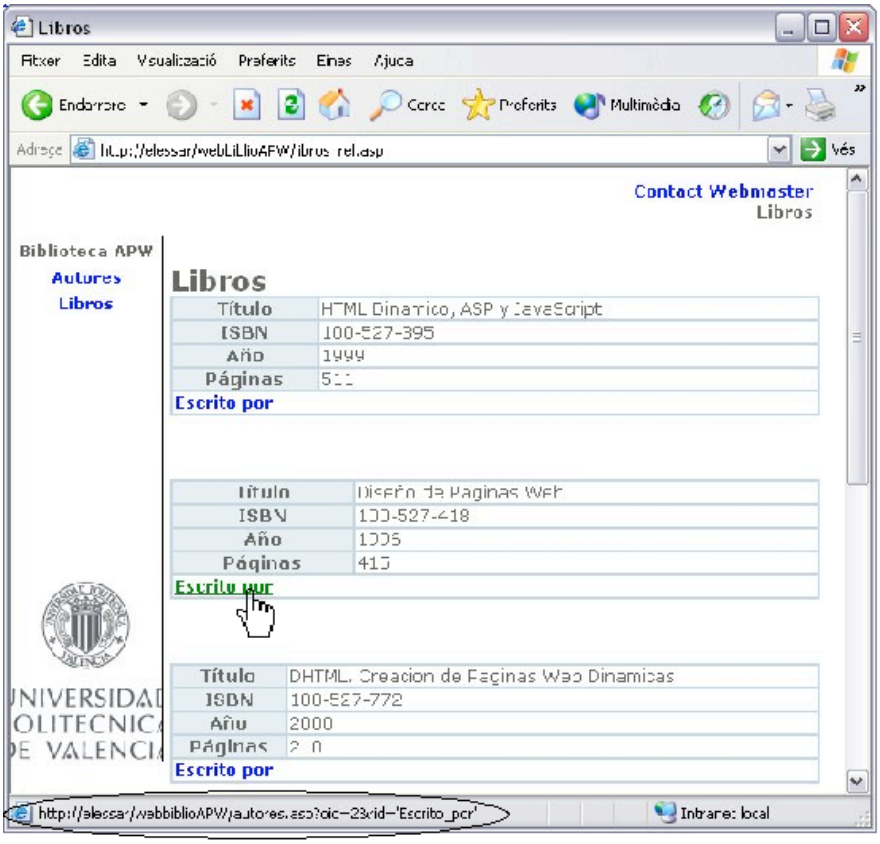


Figura 4.16 – Implementació d'una Relació de Contexte de Relació

a definir un Enllaç de Servei, només cal etiquetar l'operació navegacional amb el [contexte-destí] de la navegació, com si d'una relació de contexte es tractara.

La Figura 4.17 a la pàgina següent mostra un exemple d'enllaç de servei en la classe principal que defineix una navegació al contexte destí [Basquet de la compra] que es produirà cada vegada que l'operació *comprar()* s'active en aquest contexte.

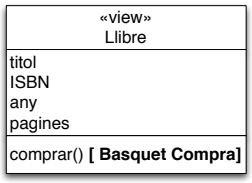


Figura 4.17 – Enllaç de Servei

4.5.5 Subsistemes Navegacionals

Un **Subsistema Navegacional** és un tipus de node dins un mapa navegacional que té com a objectiu organitzar l'estructura navegacional de l'usuari. Aquest subsistema defineix un *espai navegacional* que conté altres nodes navegacionals que sols són accessibles quan es navega dins el subsistema. Atenent a aquesta definició, dins un espai navegacional (subsistema) podran haver més subsistemes.

La Figura 4.18 a la pàgina següent mostra el que seria el *Subsistema Directori* al portal web del DSIC [39]. Ll seleccionar l'enllaç *Directori* es desplega el conjunt d'enllaços a nous continguts dins d'aquest subsistema (*Administració*, *Becaris i Col·laboradors*, *Consergeria*, *Professors*, *Tècnics* i *Sede DSIC*).

Intuitivament podriem considerar que un subsistema navegacional representa un *sub-mapa navegacional*, és a dir, un mapa navegacional dins un mapa navegacional major. L'objectiu del subsistema és el d'*encapsular* els nodes navegacional que conté. Una implementació habitual d'un subsistema és fer que els nodes continguts sols siguen visibles quan s'entre al subsistema, estant ocults en qualsevol altra situació.

Un *Subsistema Navegacional* es representa gràficament igual que un contexte navegacional (emprant un paquet d'UML), però estereotipat amb la paraula reservada ≪Subistema≫. La Figura 4.19 a la pàgina 105 mostra el mapa navegacional del DSIC, on es pot veure l'especificació de

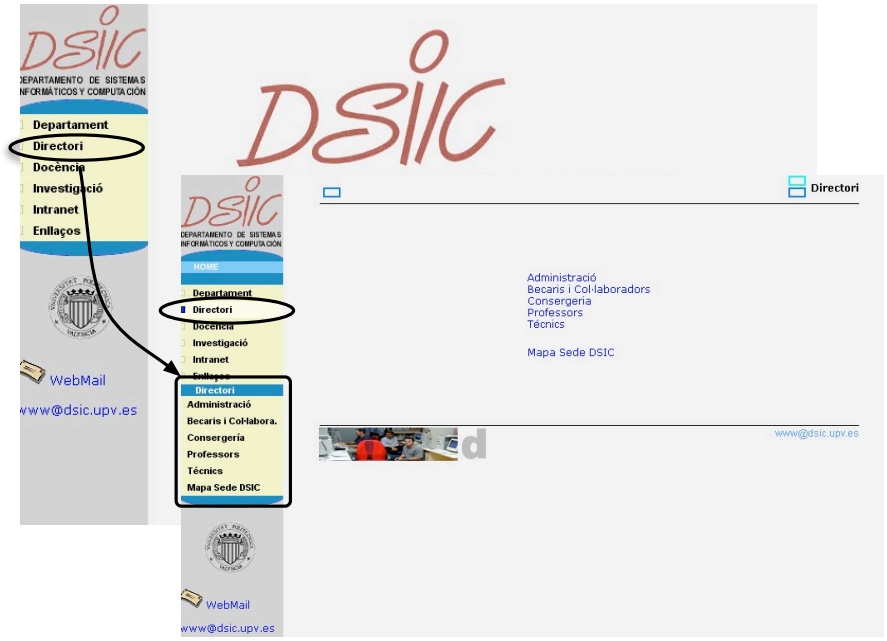


Figura 4.18 – Pàgina Web exemple de Subsistema

nodes subsistema, en concret el subsistema *Directori*.

Des d'un punt de vista formal, un mapa navegacional es representa mitjançant un graf navegacional. Els subsistemes navegacionals permeten la definició de hiper-grafs, de manera que alguns nodes del graf són a la seua vegada grafs navegacionals.

És a dir, un subsistema navegacional té una dualitat de comportament: d'una banda és un graf-mapa navegacional, i d'una altra banda és un node navegacional (que defineix un contingut format per altres nodes).

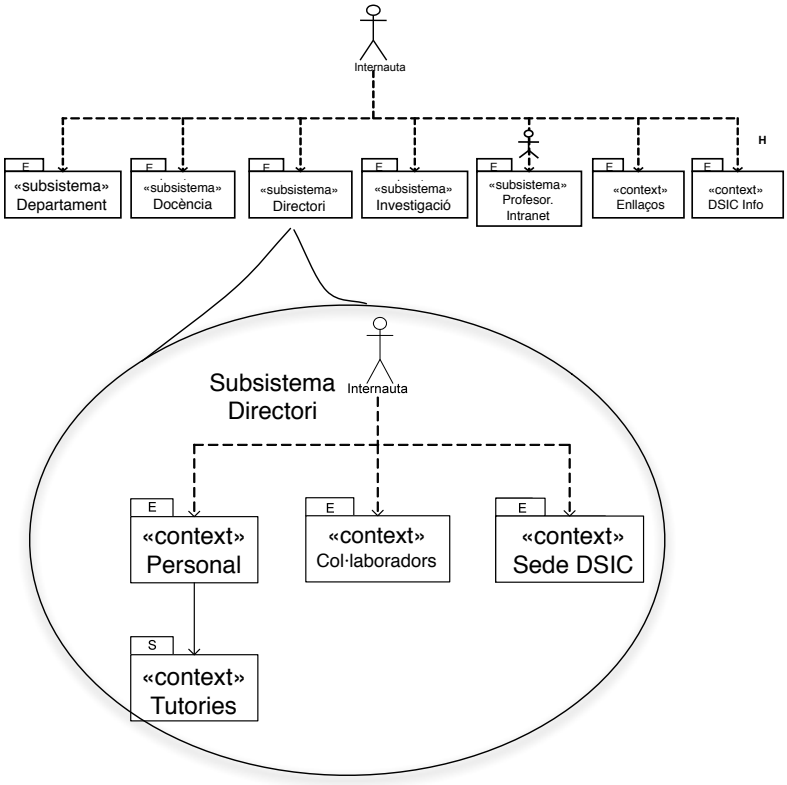


Figura 4.19 – Mapa Navegacional del DSIC

4.5.6 Usuari Conectat

La primitiva **Usuari Conectat** permet accedir a l'usuari que està conectat a l'aplicació en temps d'execució i emprar aquesta informació per establir alguna política de personalització o restricció d'accés en funció de l'usuari concret i no sols pel seu tipus d'usuari.

Sol emprar-se per establir les condicions de filtrat en les classes navegacionals. Es representa com *#usuari#*, sent *usuari* el tipus d'usuari sobre el que es defineix. Així, si es tractara d'un usuari de tipus *Bibli-*

otecari, per personalitzar la informació en base al bibliotecari contectat, s'empraria el terme *#Bibliotecari#* per fer-ne referència. Per a veure informació més detalla, veure la Secció 6.4 del Capítol 6.

4.5.7 Operacions de Sessió

Primitiva de modelatge que ens permet controlar l'execució de certes tasques (operacions) quan un usuari es conecta al sistema. Hi ha dos moments en els que es poden llançar operacions: executar un conjunt d'operacions funcionals quan l'usuari inicia una sessió d'interacció amb el sistema, o bé executar-les quan acaba la sessió.

La sintaxi per expressar aquest requeriment és la següent:

- Operacions d'inici de sessió: #Classe.Operacio#
- Operacions de fi de sessió: ##Classe.Operacio##

Aquestes operacions s'associen al mapa navegacional, i poden definir-se tantes com siga necessari.

La Figura 4.20 a la pàgina següent mostra l'especificació del mapa navegacional d'un *Internaut* de manera que, cada vegada que es conecte al sistema, el sistema executa les operacions d'inici de sessió *Internaut.IntCreate* i *Internaut.BEGINSESSION*. Quan surt de l'aplicació, el sistema executa les operacions de fi de sessió *Internaut.ENDSESSION* i *Internaut.IntDestroy*.

Amb aquestes operacions es pot aconseguir que el sistema s'actualitze amb informació de l'usuari, podent enregistrar accessos, gestionar recursos. Un cas típic és el que es dona en aplicacions de comerç electrònic on es desitja que l'usuari tinga en tot moment un basquet de la compra, de manera que cada vegada que accedisca se li inicialitze, i aquest s'esborre a l'acabar la sessió. Aquest tipus de requeriment podria representar-se fàcilment amb aquestes operacions de sessió.

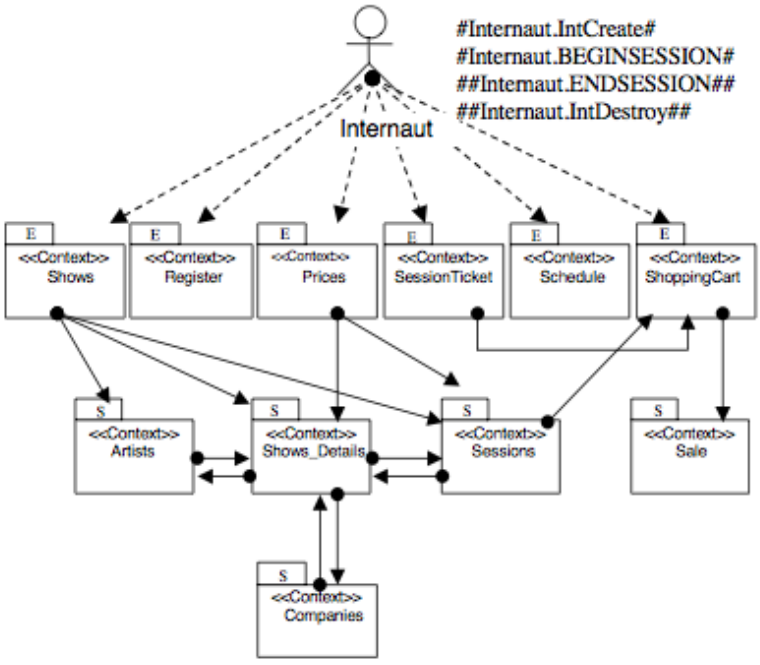


Figura 4.20 – Exemple d’Operacions de Sessió

4.5.8 Navegació entre Mapes Navegacionals

De vegades és necessari que un usuari tinga accés a un node d’un altre usuari, prèvia identificació com a aqueix usuari. Aquesta primitiva permet definir una navegació a un node que pertany al mapa d’un altre usuari. És per açò que també se la coneix com “Navegació amb canvi d’Usuari”.

El comportament definit per aquesta primitiva és el següent:

- el sistema li demanarà a aquest usuari que s’identifique com l’usuari en el que està definit el node destí. En cas que la identificació siga correcta, li permetrà l’accés al contingut del node especificat

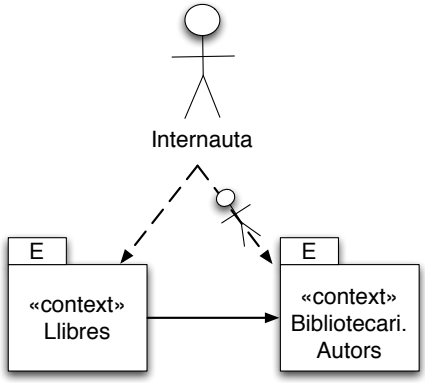


Figura 4.21 – Navegació amb canvi d'usuari

- si l'identificació ha sigut correcta, el sistema es comportarà com una navegació "normal"
- després de la navegació, l'usuari es queda identificat al sistema com l'usuari destí, podent continuar navegant pel mapa d'aquest

Hi ha dues maneres de definir una navegació entre mapes navegacionals:

- Definint una relació de contexte, especificant al seu atribut de contexte un node que pertany al mapa navegacional d'un altre usuari. Per a remarcar aquest fet, s'especificarà a l'atribut navegacional el node destí prefixat amb el nom de l'usuari propietari del mapa on està definit aquest node.
- "Important" un node navegacional d'exploració al mapa actual. D'aquesta manera es donarà accés a aquest, però no es podrà "redefinir" el seu contingut.

4.6 Conclusions

En aquest capítol s'ha presentat el *Model de Navegació* d'OOWS. Aquest model extèn la fase de modelatge conceptual permetent descriure els requeriments del sistema d'informació que tenen a veure amb la "navegabilitat"[7] pel sistema.

Aquest Model Navegacional es construeix en dues fases, com és habitual en les aproximacions que segueixen els principis de l'Enginyeria Web:

1. En una primera fase es representen els conceptes més generals i estructurals de la navegació, mitjançant un *Mapa Navegacional* on es proposen els nodes i enllaços entre nodes.

2. En una segona fase es detalla el contingut d'aquestos nodes o *Contextes Navegacionals* mitjançant un conjunt de primitives navegacionals que permeten extreure informació del sistema (definint si cal restriccions sobre els conjunts de població), definir operacions que es podran executar en aquestos contextes i enllaçar la informació dels diferents contextes definint la "navegabilitat pel sistema".

Allò important d'aquest model no és sol que permet descriure les propietats navegacionals essencials de les aplicacions Web, sinò que a causa de la seua construcció, està fortament relacionat amb la resta de models que formen part del model conceptual del sistema (més en concret, amb el diagrama d'estructura d'informació o diagrama de classes). El fet de basar-lo amb OO-Method ha permés entendre de manera unívoca les implicacions de les primitives de modelatge que aquest empra, evitant certs problemes d'ambigüitat i extensió d'altres aproximacions, com ara l'UML.

[7] Aquesta navegabilitat no és la mateixa que la propietat *Navegabilitat* que s'empra en algunes aproximacions per caracteritzar relacions d'Associació als Diagrama de Classes.

A causa d'aquesta simbiosi, s'ha pogut donar una proposta a alguns aspectes que avuí en dia encara no tenen una solució consensuada en l'àmbit de l'enginyeria web com és la relació entre la navegació i la funcionalitat del sistema, o la relació entre els continguts d'informació dels nodes navegacionals i l'estructuració conceptual de la informació. Per exemple, al cas de la relació entre la navegació i la funcionalitat, alguns autors defineixen operacions sobre enllaços navegacionals, descrivint l'efecte d'aquesta navegació sobre l'estructura d'informació del sistema. Amb OOWS, açò està clarament separat: hi existirà l'operació definida al Diagrama de Classes OO-Method, expressant les seues implicacions mitjançant els Models Dinàmic i Funcional. Al Model Navegacional es podrà definir una Enllaç de Servei, que defineix una navegació i provoca l'execució d'una operació del sistema.

Capítol 5

Model d'Usuaris

5.1 Introducció

L'Enginyeria Web promou l'adaptació dels mètodes de producció de programari per tal d'introduir els conceptes i requeriments essencials de les aplicacions web. Un d'aquestos requeriments és el de gestionar correctament els (potencials) usuaris d'una aplicació web.

Malgrat açò, OO-Method sí que introdueix els usuaris al seu diagrama de classes, fent-los explícits. L'objectiu és doble: (1) d'una banda permet representar-los com estructures d'informació (persistents), especificant les seues propietas i relacions amb altres entitats del Diagrama de Classes i (2) definir quina visibilitat té sobre la resta de classes del sistema, indicant quins atributs, relacions i operacions té permís per explorar i executar.

Així, l'objectiu d'aquest capítol és introduir el **Diagrama d'Usuaris** com un mecanisme que permeta tenir en compte als tipus d'usuaris d'una manera clara i explícita, i així poder gestionar-los i representar els seus requeriments navegacionals.

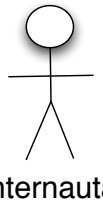


Figura 5.1 – Representació gràfica d'un Usuari

5.2 El Diagrama d'Usuaris

El **Diagrama d'Usuaris** és una eina que permet definir explícitament els tipus o perfils d'usuari que podran accedir al sistema i les relacions entre aquestos usuaris. Cada aplicació web haurà de tenir el seu propi Diagrama d'Usuaris.

Un *tipus d'usuari*[1] es representa gràficament mitjançant un actor UML, com es pot veure a la Figura 5.1

Per construir un diagrama d'usuaris devem seguir els següents passos:

1. Detectar i Definir els tipus d'usuaris
2. Asignar un mode d'accés a cada tipus d'usuari
3. Detectar i Definir les relacions entre els usuaris

Per *Detectar i Definir els tipus d'usuaris* normalment observarem els requeriments del sistema i extraurem aquells referents a qui pot utilitzar el sistema.

Suposem una Biblioteca on-line on hi ha: Internautes que exploren el catàleg, Bibliotecaris que gestionen el catàleg, Socis que poden realitzar prèstec, el/els Director/s que gestionen la infraestructura de la Biblioteca

[1] Normalment l'anomenarem directament *Usuari*.

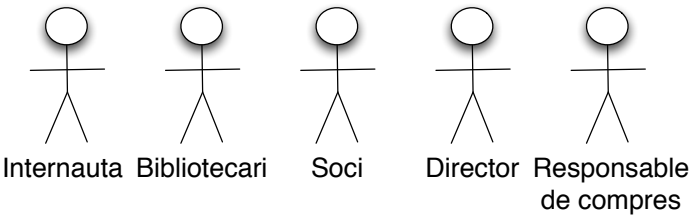


Figura 5.2 – Diagrama d'Usuaris de la Biblioteca online: Detecció dels Usuaris

i els Responsables de Compres que s'encarreguen de comprar el material bibliogràfic.

El primer pas, doncs, per constuir el Diagrama d'Usuaris d'aquesta Biblioteca seria extraure i explicitar els usuaris. Açò pot veure's a la Figura 5.2.

Atenent al tipus d'accés o "permís de conexió" que poden tenir els usuaris de cara al sistema, es distingeixen 3 tipus d'usuaris: *anònims*, *registrats* i *sense permís*.

5.2.1 Usuaris Anònims

Un usuari **Anònim** (Figura 5.3 a la pàgina següent) no necessita identificar-se al conectar-se al sistema. Símplement tenen accés. Per aquesta naturalesa, açò fa que siga molt habitual que els tipus d'usuaris marcats com *anònims* tinguen uns permisos d'accés al sistema molt reduïts. És molt recomanable que al tipus d'informació i funcionalitat que puguen accedir siga "no-sensible", "no-crítica", "no-personal", etc., ja que el sistema no pot saber qui hi ha al darrere i seria "responsable" d'algunes accions.

Com que d'aquestos usuaris no es té més informació que el fet de saber que són anònims, no es podrà aplicar cap política de personalització dinàmica (veure Secció 6.4 del Capítol 6).

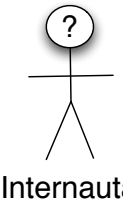


Figura 5.3 – Usuari de tipus anònim

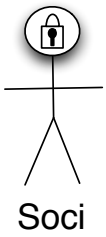


Figura 5.4 – Usuari de tipus registat

5.2.2 Usuaris Registrats

Un usuari **Registrat** (Figura 5.4) necessita obligatòriament identificar-se al conectar-se al sistema, de manera que aquest sàpiga, no sols quin tipus d'usuari és, si nó també quin usuari concret. Avuí en dia, aquestes identificacions solen produir-se proporcionant un identificador d'usuari ("*login*") i una clau d'accés. Aquestos són els tipus d'usuaris amb més responsabilitats amb el sistema, i als que se'ls associa accés a informació i funcionalitat "sensible", "crítica", "personal", etc.

A aquestos usuaris si se'ls pot aplicar polítiques de personalització dinàmica (Secció 6.4 del Capítol 6) i és molt habitual establir condicions de filtrat sobre la població (veure Secció 4.5.6 del Capítol 4) en base a la seua informació.

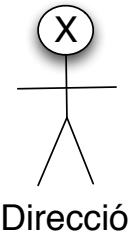


Figura 5.5 – Usuari de tipus "Sense Permís"

5.2.3 Usuari "Sense Permís"

Un usuari **"Sense Permís"** (Figura 5.5) no pot accedir al sistema. Són usuaris "virtuals" o "sintètics" creats a propòsit (no existeixen en realitat) per tal de: (1) expressar responsabilitats comuns entre usuaris, (2) plasmar una jerarquia funcional (per exemple, en una empresa, Departaments, Seccions, Unitats, etc.) i (3) organitzar i estructurar els usuaris del sistema.

Aquest tipus d'usuari només té sentit quan parlem de relacions entre usuaris (veure la Secció 5.3). Aquest usuari representa el "factor comú" de la vista navegacional compartida entre diferents usuaris.

5.2.4 Exemple de Diagrama d'Usuaris

Seguint amb l'exemple de la Biblioteca on-line, podriem afinar el seu Diagrama d'Usuaris introduint la informació relacionada amb el mode d'accés de cada tipus d'usuari. La Figura 5.6 a la pàgina següent mostra l'especificació d'aquest requeriment. Com es pot observar, s'ha definit l'Internauta com un usuari *anònim*, ja que per tal d'explorar el catàleg no anem a obligar que es registre, mentre que tota la resta d'usuaris són *registrats* ja que és important saber qui són a l'hora de realitzar les accions a què estan capacitats.

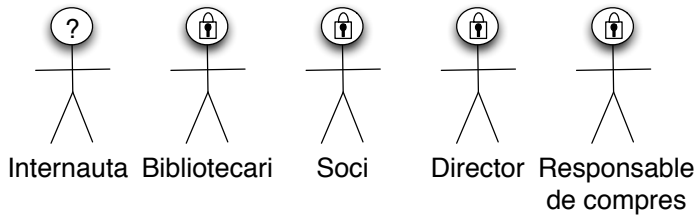


Figura 5.6 – Diagrama d'Usuaris de la Biblioteca online: Mode Accés

5.3 Relacions entre Usuaris

És molt habitual que diferents usuaris tinguen accés al mateix contingut d'informació i/o compartisquen responsabilitats de cara al sistema (puguen realitzar operacions i accions comunes). Què fem aleshores? Repetim l'especificació? Tractem de fusionar allò comú?

Aquest problema no és nou. Per exemple, quan estem descrivint les estructures d'informació del sistema mitjançant un Diagrama de Classes, ens pot passar que detectem que dues (o més) classes comparteixen tota o part de les estructures d'informació. En aqueix moment, el que es fa es definir una *jerarquia d'herència* entre les classes, de manera que es crea una classe "pare" que conté allò comú i classes "filles" que hereten les propietats del pare (herència estructural). Aquestes propietats estructurals són, bàsicament, els atributs, les operacions i les relacions estructurals amb altres classes (associació-agregació-composició i especialització-generalització).

La idea, en aquest punt, és seguir el mateix camí ací, a la definició dels usuaris. Quan hi ha dos (o més) usuaris que comparteixen part de l'accés al sistema (que en OOWS es defineix mitjançant el mapa navegacional, com s'ha vist a la Secció 4.4 del Capítol 4), es crearà un usuari "pare" que continga allò comú i una sèrie d'usuaris "fills" que hereten les seues propietats. Es defineix així, intuitivament el concepte

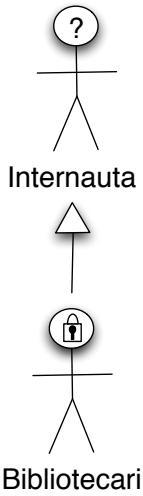


Figura 5.7 – Especialització entre Usuaris

d'**especialització navegacional**. A la Secció 6.5 del Capítol 6 es pot obtenir més informació sobre aquest concepte i les seues repercusions.

La Figura 5.7 mostra a dos usuaris amb una relació d'especialització navegacional definida entre ells. Cal notar que, encara que la representació gràfica de l'herència siga la mateixa que en un Diagrama de Classes, el significat és diferent ja que s'està aplicant en el contexte d'un Diagrama d'Usuaris i entre tipus d'usuaris, i no entre classes.

Per construir el Diagrama d'Usuaris només cal saber que si especialitzem un usuari a partir d'un altre, l'usuari especilitzat:

- heretarà el mapa de l'usuari base, i per tant l'accés als nodes i enllaços navegacionals definits
- podrà modificar el seu mapa, afegint, modificant o llevant accés a nodes i enllaços

Al final, un Diagrama Navegacional podrà tenir una estructura

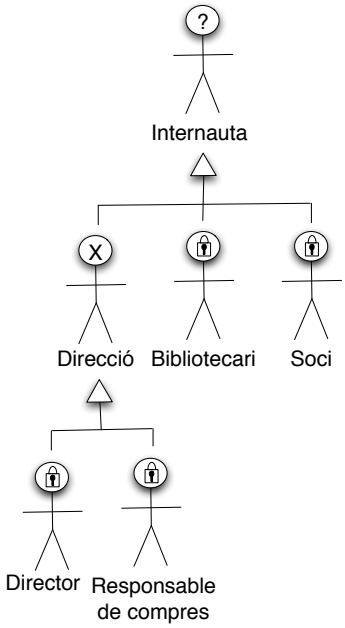


Figura 5.8 – Diagrama d'Usuaris Biblioteca on-line: Relacions entre Usuaris

jeràrquica si tots els seus usuaris estan relacionats (no és obligatori), sempre que es respeten algunes consideracions:

- els usuaris sense permís no són usuaris finals, és a dir, sempre han d'estar especialitzats en altres usuaris
- un usuari anònim no pot especialitzar-se d'un usuari registrat, però sí d'altre usuari anònim o d'altre usuari "Sense Permís" (ni directa, ni indirectament)

Tenint en compte aquestes relacions entre usuaris, el Diagrama d'Usuaris de la Biblioteca podria afinar-se. Així, per exemple, per indicar que tots els usuaris de la Biblioteca poden explorar el catàleg (almenys de la mateixa manera que un Internauta, encara que després puguen

afegir més funcionalitat), el que podem fer és fer que tots els usuaris hereten de l'Internauta (directa o indirectament). A més, el Director i el Responsable de Compres tenen accés comú i altre diferenciat. Per tant, podem crear un usuari virtual "Sense Permís" Direcció, que reunisca aquesta informació i que després s'especialitze. La Figura 5.8 a la pàgina anterior mostra com podria quedar aquest Diagrama d'Usuaris complet, amb les relacions entre usuaris definides.

5.4 Conclusions

En aquest capítol s'ha proposat l'ús d'un *Diagrama d'Usuaris* explícitament en la proposta OOWS. L'objectiu és clar: que la definició de les interfícies web siguen guiades per l'anàlisi previ de l'"audiència" o usuaris finals de l'aplicacions. Cada usuari abstrau una "vista" del sistema, orientada a aconseguir que l'usuari realitze les seues tasques o responsabilitats de cara al sistema. Aquesta vista és definida mitjançant el Model Navegacional vist al Capítol 4.

A pesar que en OO-Method els usuaris es representen directament en el Diagrama de Classes (com part de la societat en l'estructura d'informació), per descriure les propietats navegacionals cal capturar informació addicional relacionada amb la interacció amb el sistema.

D'aquesta manera, i prèvia la construcció del Model Navegacional, es realitza la descripció dels usuaris del sistema i es detecten les possibles relacions entre ells que descriuen les parts del sistema que són compartides entre aquestos (que es pot veure com responsabilitats comunes entre usuaris de cara al sistema).

Aquest model, a banda, serveix com a base per a realitzar les extensions d'adaptativitat i personalització que es veuen a [101].

Capítol 6

Conceptes Avançats

6.1 Introducció

Al Capítol 3 s'ha presentat una proposta de procés de desenvolupament d'aplicacions web basat en modelatge conceptual i al Capítol 4 s'ha presentat el conjunt de primitives conceptuals bàsiques per representar el concepte de navegació en les etapes de modelatge conceptual. En aquest capítol es presenten una sèrie de millores al mètode presentat (tant en expressivitat com en el seu procés d'especificació) orientades a donar un suport a aspectes importants inherents a les aplicacions web.

La proposta que es fa d'aquestes millores es fa atenent a dos criteris essencials: (1) incorporar cada element en el seu nivell d'abstracció pertinent i (2) trobar el "moment" (dins el procés de desenvolupament) més adient per descobrir i descriure aquestes propietats.

Aquest capítol està dividit en les següents seccions:

- A la Secció 6.2 es proposa una estratègia diferent per especificar *mecanismes d'accessibilitat* dins el sistema, aplicant principis metodològics que permeten detectar quan aquestos mecanismes són necessaris o recomanables.

- A la Secció 6.3 s'introdueix una primitiva que permet descriure l'*agregació de continguts* en les aplicacions Web, molt habitual als portals web contenidors. En aquests sol apareixer de gran quantitat d'informació no relacionada entre sí, però que és combinada per construir les pàgines web de l'aplicació.

- A la Secció 6.4 es descriuen les bases per a la *personalització* d'aplicacions Web amb OOWS, atenent al concepte més general d'*Adaptativitat*.

- A la Secció 6.5 es defineix el concepte d'*especialització navegacional* que permet representar relacions entre requeriments navegacionals d'usuaris i definir les influències d'aquestes relacions. Entre altres, aconseguir una reutilització en l'especificació navegacional.

6.2 Mecanismes d'Accés a la Informació

L'objectiu d'un **mecanisme d'accés a la informació** és facilitar la búsqueda d'informació dins el sistema. En les aplicacions web actualment implementades i en els mètodes de desenvolupament d'aplicacions que segueixen els principis de l'Enginyeria Web, s'empren diferents d'aquestos mecanismes [107]: indexs, filtres, visites guiades, etc. Cadascun d'aquestos mecanismes proveeixen a l'usuari d'informació que li permet fer una cerca més eficient d'allò que realment està buscant.

Aquests mecanismes són emprats en les etapes de descripció de la navegabilitat pel sistema incloent-los en el "graf navegacional" principal que descriu l'aplicació web. Açò fa que aquest tipus de mecanismes siguen considerats com "ents de primer nivell", comparables als nodes que especifiquen informació. És a dir, són un tipus més de nodes del graf, l'objectiu del qual és proporcionar accés al contingut d'algun altre node.

Des d'un punt de vista de procés de construcció d'especificacions conceptuals, l'aproximació presentada en aquesta tesi fa una estratègia diferent, ajornant el moment d'especificació d'aquests mecanismes per aconseguir bàsicament dos avantatges: (1) simplificar el mapa (o graf) navegacional, especificant només nodes que defineixen contingut, i (2) aprofitar al màxim el coneixement recollit en la construcció d'aquest mapa navegacional i poder predir on (en quins nodes) i en quines circumstàncies de navegació serà necessari que s'activen aquests mecanismes. Tot açò serà revisat a la secció 6.2

6.2.1 Problemàtica

Els contextes navegacionals defineixen una recuperació de la població de les classes navegacionals definides a les Unitats d'Abstracció d'Informació (UAI). Es defineix la **cardinalitat d'una UAI** com la cardinalitat de la població de la UAI ó dit d'una altra manera, com el número d'instàncies que satisfan la vista de la UAI en un instant determinat.

De vegades, la cardinalitat d'una UAI és elevada i resulta difícil de gestionar per l'usuari, arribant a deixar de ser usable [31]. Per tal d'ajudar als usuaris a cercar per aquesta quantitat d'informació, s'empren els mecanismes d'accés.

Per tal d'entendre bé el problema, s'explicarà amb el següent exemple d'una biblioteca o tenda on-line de venda de llibres. Es plantejarà la problemàtica sobre aquesta, i en les següents subseccions es resoldrà aplicant els mecanismes que es proposen.

Suposem que tenim una *tenda on-line de llibres*. Aquests llibres estan catalogats per autors i per temàtiques, de manera que per cercar un llibre es pot accedir directament al catàleg dels llibres, o bé fer-ho per mig del seu autor o temàtica. Suposem també que hi ha gran quantitat de llibres a la tenda. La Figura 6.1 a la pàgina següent mostra la part del diagrama de classes que representa aquest requeriment. Com es pot

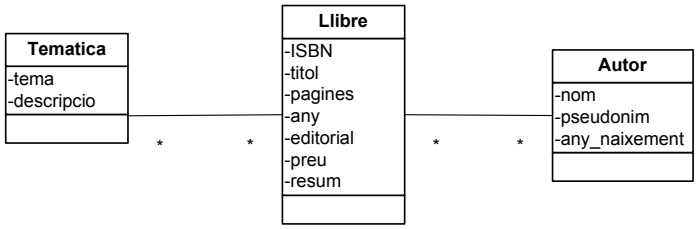


Figura 6.1 – Tenda on-line: Diagrama Classes

apreciar, un llibre pot estar relacionat amb diferents temàtiques i una temàtica pot tenir diversos llibres, en principi molts. També, un llibre pot estar relacionat amb varios autors i un autors pot tenir diversos llibres, en principi també molts.

Suposem ara que volem desenvolupar l'aplicació web basada en aquesta estructura d'informació i que permeta als usuaris internautes explorar el catàleg de llibres de les següents maneres:

1. Accedir dirèctament al catàleg dels llibres disponibles
2. Accedir a informació dels llibres a partir d'algun autor
3. Accedir a informació dels llibres d'alguna temàtica en particular

Si revisem aquests requeriments del sistema, podem observar que es tracta de requeriments de navegació en els quals s'estableix que existirà un usuari *Internauta* amb el mapa navegacional de la Figura 6.2 a la pàgina següent.

Com es pot observar en aqueixa figura, un *Internauta* pot:

- accedir al contexte *Llibres* (veure Figura 6.3 a la pàgina 126) emprant un enllaç d'exploració per explorar tot el catàleg de llibres,

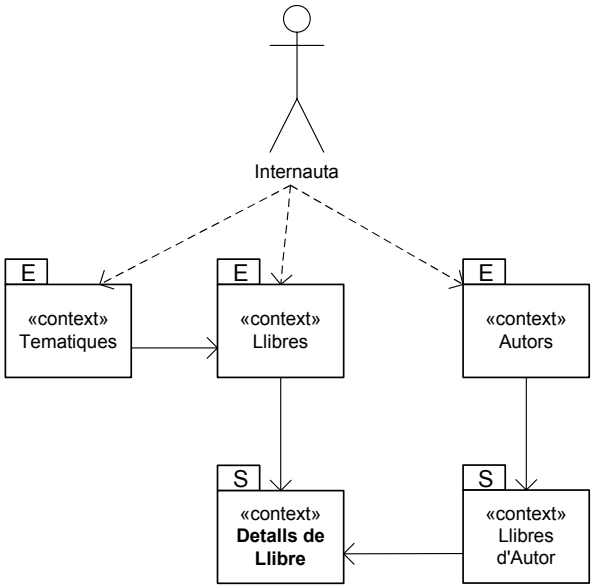


Figura 6.2 – Tenda on-line: Mapa Navegacional de l'Internauta

- accedir per exploració al contexte *Autors* per veure els autors del sistema i al seleccionar-ne un d'autor, navegar al contexte *Llibres* emprant un enllaç de seqüència per veure els llibres escrits per aquest autor

- accedir per exploració al contexte *Temàtiques* per veure els autors del sistema i al seleccionar una temàtica, navegar al contexte *Llibres* emprant un enllaç de seqüència per veure els llibres de la matèria d'aquesta temàtica

Si tenim aquesta estructura navegacional, se'ns podrien plantejar els següents problemes:

- Quan naveguem al contexte *Llibres* per exploració, què esdevindria si hi ha gran quantitat de llibres al catàleg?

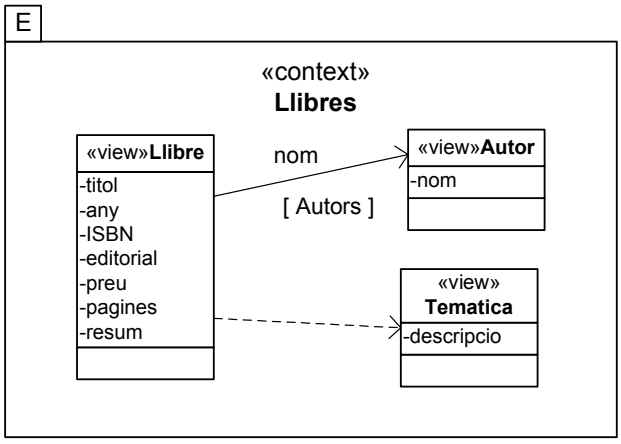


Figura 6.3 – Tenda on-line: Contexte Llibres

- Quan naveguem al contexte *Autors* per exploració, què esdevindria si hi ha gran quantitat de autors al catàleg?
- Quan naveguem al contexte *Temàtiques* per exploració, què esdevindria si hi ha gran quantitat de temàtiques al catàleg?
- Quan naveguem per seqüència al contexte *Llibres* des del contexte *Autors*, què esdevindria si un autor ha escrit molts llibres?
- Quan naveguem per seqüència al contexte *Llibres* des del contexte *Temàtiques*, què esdevindria ha molts llibres de la mateixa temàtica?

La resposta a totes les preguntes anteriors és que, si no s'especifica res, el sistema li oferiria a l'Internauta una gran quantitat d'informació. És raonable pensar que açò podria provocar que l'usuari se sentira sobrepassat amb la gran quantitat d'informació i se li ferai dificultós el trobar el/s llibre/s que està buscant.

Per minvar aquests problemes, al món de l'Enginyeria Web venen

emprant-se uns mecanismes que faciliten un accés més ràpid a la informació buscada per l'usuari. Aquests mecanismes són típicament tres: els índexos, els filtres i les visites guiades [107].

En aquest sentit, les aproximacions més rellevants de desenvolupament d'aplicacions hipermedials (OOHDM, WebML, UWE, etc.) empren aquests mecanismes *quan s'està construint el graf navegacional.* D'aquesta manera es produeixen en una sèrie de desavantatges:

- L'analista ha de tenir en compte el disseny de l'accés a les dades mentre construeix la navegabilitat pel sistema. Aquesta hauria de basar-se essencialment en les relacions estructurals de dades, i no centrar-se amb altres detalls irrellevants a aquest nivell. De fet, podria arribar-se a la conclusió que el fet d'introduir aquests mecanismes influeix en la navegació pel sistema (veure revisió del concepte de navegació en la Secció 4.2 del Capítol 4). Aquests mecanismes no impliquen navegació conceptual, i per tant, no haurien d'aparèixer com a nodes dins del graf navegacional.

- Els grafs navegacionals es tornen massa sobrecarregats. Apart d'aparèixer els nodes navegacionals que descriuen les vistes dels usuaris, aparèixen els mecanismes d'accés que s'intercalen en la navegació entre els nodes. Açò esdevé en una complexitat addicional del graf navegacional.

- L'analista no pot assegurar que empra tots els mecanismes d'accés necessaris ni que no especifica alemnys els més pertinents. Depén del seu criteri únicament que veja totes les navegacions que puguen necessitar la incorporació d'algun mecanisme per gestionar la gran quantitat d'informació.

- L'analista ha de duplicar l'especificació de mecanismes d'accés quan vulga arribar a un mateix contexte des de diferents nodes.

> Així, per exemple, si es vol navegar al node *Llibres* a partir de l'autor o de la temàtica, caldria crear dos mecanismes diferents. En OOHDM, per solucionar aquest problema poden parametritzar aquestos mecanismes, però les postres hi són necessaris els dos mecanismes.

L'objectiu serà doncs incorporar aquests mecanismes d'accés de manera que eviten aquests problemes esmentats i que reaprofiten al màxim el coneixement especificat, de manera que siga possible detectar, a partir d'un graf navegacional d'alt nivell, en quins punts caldrà actuar per introduir els mecanismes d'accés.

6.2.2 Els Índexs

Un **índex** és una estructura que proporciona un accés "indexat" a la població sobre la que es defineix. Els índexs creen una llista d'informació resumida, atenent a una propietat. A causa que un índex defineix com accedir a un conjunt de dades, pareix sensat pensar que aquests van associats al concepte de conjunt de visualització d'OOWS, o *UAI*.

Dit d'una altra manera, un índex crea una *classe d'equivalència* d'una "població" de dades[1]. Mitjançant aquesta operació es poden crear particions del conjunt de visualització atenent a la propietat que defineix la classe d'equivalència. L'objectiu és doncs crear classes d'equivalència de cardinalitat molt menor que la cardinalitat de la població global. El conjunt dels *representants* de les classes d'equivalència és el que constitueix l'índex[2]. Al seleccionar una de les classes d'equivalència, s'obtindrà com a resultat la "població" d'aqueixa classe.

És a dir, la propietat fonamental que defineix un índex és la seua **propietat d'indexació**. Si la propietat s'estableix sobre un atribut de

[1] Anomenem població al conjunt de les dades que s'han de visualitzar emprant una propietat per definir-se.

[2] Un índex es representa normalment com una llista de valors.

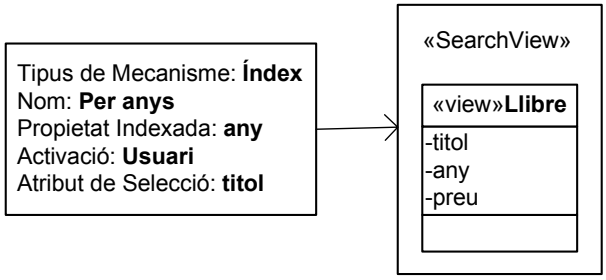


Figura 6.4 – Tenda on-line: Especificació d'un Índex

la classe directora en la que es defineix l'índex, aquest s'anomena **índex d'atribut**. Si la propietat pertany a qualsevol classe complementària, l'índex s'anomena **índex de relació**.

L'*activació* de l'índex pot ser de dues maneres: iniciada per l'usuari que explícitament selecciona l'índex per a que s'active, o pel sistema. En aquest segon cas, l'usuari veurà directament l'índex ja format i no necessitarà activar-lo explícitament.

Quan un índex s'activa, es crea una llista amb tots els possibles valors per a la/les propietat/s indexada/es. Si se'n selecciona algun d'aquestos valors, es recuperaran totes les instàncies de la classe directora que tinguen aquest valor i es veurà el resultat és una **vista de resultats**. Aquesta vista de resultats descriu la informació que se li proporcionarà a l'usuari per tal que puga triar quina/es instància/es li són del seu interés. Per fer-ho, caldrà especificar una recuperació d'informació en una UAI estereotipada amb la paraula reservada ≪SearchView≫ (veure Figura 6.4.)

Associada amb la vista de resultats està la propietat *atribut de selecció*, que indica l'atribut que s'emprarà per seleccionar les instàncies de la vista. Si no se n'especifica cap, la selecció es realitzarà de manera independent a la informació mostrada en la vista[3].

[3]Per exemple, mitjançant un botó de selecció associat a cada instància trobada.

Amb açò permetem reduir la quantitat d'informació que es recupera atenent a una propietat concreta. A més més, no se li demana a l'usuari que introduïsca el valor de la propietat que està buscant: el sistema li proporciona totes les possibilitats diferents que existeixen mitjançant els representants de cada classe d'equivalència que conformen l'índex.

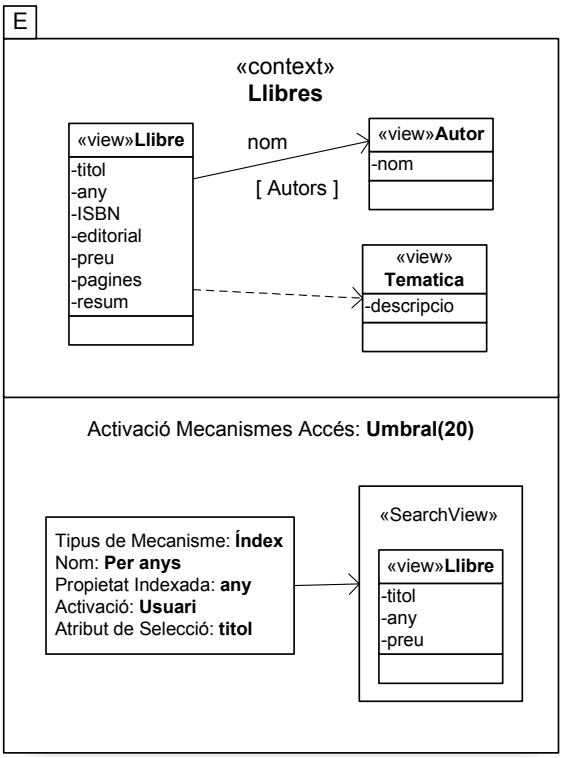


Figura 6.5 – Tenda on-line: Contexte Llibres amb Índex

Suposem que volem accedir als llibres de la tenda on-line de llibres en funció de l'any de publicació. Allò que ens agradaria és que el sistema ens proporcionara una llista de tots els anys als que hi ha alguna publicació,

i poder seleccionar un d'aquestos anys per veure totes les publicacions que té. Per especificar aquest requeriment, només caldria definir un *index d'atribut* sobre la propietat *any* d'un *Llibre*. La Figura 6.5 a la pàgina anterior mostra el contexte *Llibres* amb l'especificació de l'índex *Per anys*.

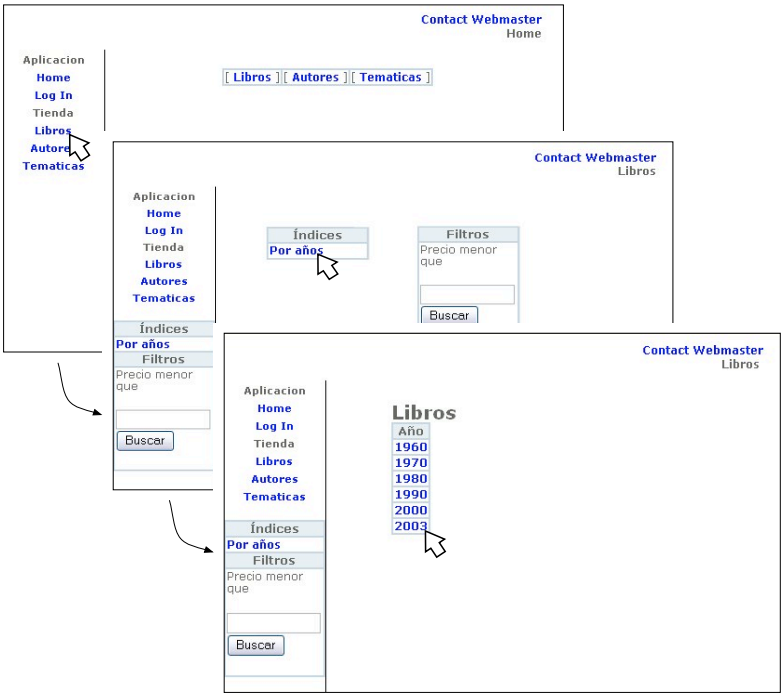


Figura 6.6 – Tenda on-line: Ús d'un índex per a Llibres

A partir del Mapa Navegacional i el Contexte amb l'especificació de l'índex, s'està descrivint el següent ús de l'índex (Figura 6.6). Primer, l'Internauta navega al contexte *Llibres* a partir de la pàgina per defecte. Com que l'activació dels mecanismes s'ha establert a un umbral de 20 instàncies (veure Secció 6.2.5) i suposem que hi ha més de 20

llibres, el sistema activa l'ús dels mecanismes i no mostra la informació del contexte. Quan l'usuari activa l'índex explícitament al segon pas, (Activacio: Usuari), el sistema li proporciona el llistat dels anys als que hi ha publicacions al sistema.

Figura 6.7 – Tenda on-line: Selecció del llibre

Per últim, l'Internauta obté el llistat de tots els llibres que satisfan l'índex, segons la *vista de resultats* especificada. Després de seleccionar un d'aquestos llibres, obté la informació especificada al contexte (veure Figura 6.5 a la pàgina 130).

6.2.3 Els Filtres

Un **filtre** és un mecanisme que permet restringir la població a recuperar en funció d'alguna propietat o condició de filtrat. És l'usuari qui estableix el valor o rang de valors que està buscant.

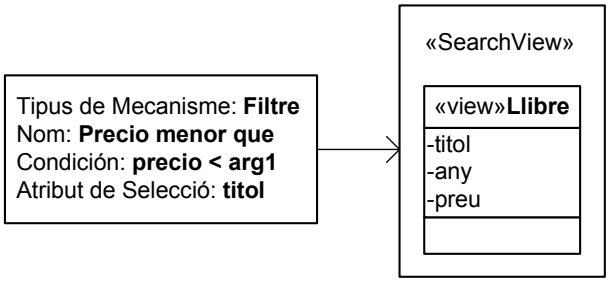


Figura 6.8 – Tenda on-line: Especificació d'un Filtre

La propietat fonamental que defineix un filtre és la seua **condició** o fòrmula. Si aquesta condició està *tancada*, obtenim un **filtre estàtic**. Si la condició està *oberta* en funció d'arguments, tenim un **filtre dinàmic**. En aquest segon cas, el sistema necessitarà que l'usuari especifique el valor dels arguments per poder concretar la fòrmula. Aquestes condicions s'estableixen sobre la població de la UAI a la que s'aplica.

Quan un filtre s'activa, mostra les instàncies del conjunt de visualització sobre el que s'aplica en una **vista de resultats**. Aquesta vista de resultats es comporta exactament que la vista de resultats d'un índex, podent inclús compartir-se.

Suposem que ara volem accedir als llibres de la tenda on-line en funció del seu preu. Volem que l'usuari puga indicar una quantitat, i que el sistema només proporcione aquells que estan sota aquest preu. Per especificar aquest requeriment, caldria definir un *filtre dinàmic* amb la següent condició sobre el *preu* del *Llibre*: $\mathsf{preu} < \mathsf{arg1}$[4]. La Figura 6.9 a la pàgina següent mostra el contexte *Llibres* amb l'especificació del filtre *Precio menor que*, incloent també l'especificació de l'índex anterior.

A partir del Mapa Navegacional i el Contexte amb l'especificació del filtre, s'està descrivint el següent ús del filtre (Figura 6.10 a la pàgina 135).

[4]arg1 representa un argument a especificar per l'usuari, amb el mateix tipus que l'atribut preu.

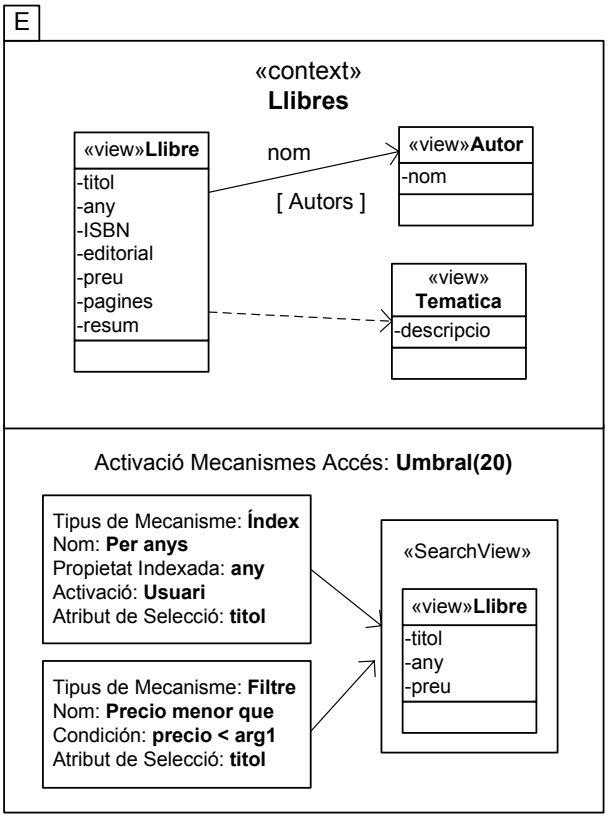


Figura 6.9 – Tenda on-line: Contexte Llibres amb Filtre

Primer, l'Internauta navega al contexte *Llibres* a partir de la pàgina per defecte. L'umbral d'activació (20) fa que no es mostre la població del contexte. L'usuari estableix el valor per a l'argument del filtre, indicant que vol recuperar aquells llibres amb preu inferior a 16 euros, i el sistema li proporciona tots aquestos atenent a la vista de resultats. L'últim pas és que l'usuari seleccione un llibre per veure la informació completa.

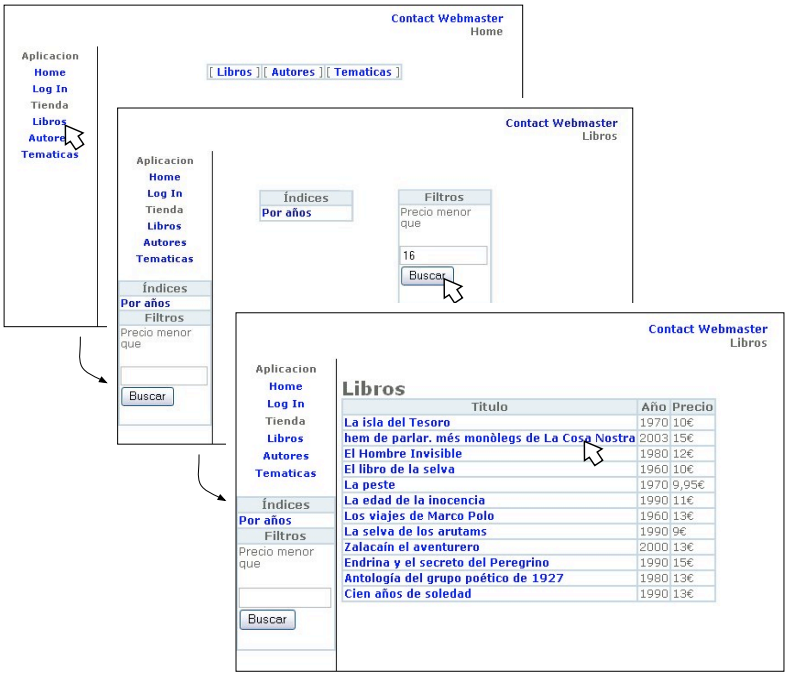


Figura 6.10 – Tenda on-line: Ús d'un filtre per a Llibres

6.2.4 Les Visites Guiades

Una **visita guiada** és un mecanisme d'accés que permet definir un "tour" per una colecció de dades. En el seu ús actual més extés, aquest mecanisme permet establir, en etapes de modelatge conceptual, una seqüència predefinida d'objectes que seran explorats. És molt corrent, doncs, que siga emprat en sistemes en els quals es coneixen les instàncies dels objectes que poblaran el sistema.

L'exemple aclaratori més habitual és dóna en sistemes hipermedials per a Museus. La definició de visites guiades permet establir una ordenació "navegacional" de les obres que aniran sent explorades.

Però, açò imposa la restricció de treballar amb instàncies, fent depenent l'especificació (model conceptual) de les poblacions que existiran en temps d'execució. Encara que aquest fenòmen ha estat discutit i tractat durant llarg temps [107], en l'aproximació OO-Method/OOWS no es permet la introducció d'aquests objectes-instància a nivell de modelatge conceptual.

Des del punt de vista de l'aproximació OOWS, el que ens interessa es poder definir una altra classe de visites guiades en les quals no es defineix a priori (en etapa de modelatge conceptual) el camí per explorar les instàncies. Una visita guiada es podria definir com un camí entre una població atenent a una ordenació per una propietat. Per exemple, una exploració de les obres pictòriques d'un gènere ordenades ascendentment per l'any de creació. Aquest tipus de requeriments es defineixen al model de Presentació (veure Capítol 7) amb les propietats de paginació d'informació i criteri d'ordenació de dades.

6.2.5 Activació dels Mecanismes d'Accés

Per acabar amb la descripció dels mecanismes d'accés, cal descriure una propietat addicional d'aquestos: l'activació a nivell de contexte. Aquesta activació determina quan un contexte ha de mostrar el seu contingut, o si requereix que per explorar els conjunts de visualització siga necessari emprar un mecanisme d'accés. Aleshores, per a cada UAI (conjunt de visualització) que hi haja dins un contexte, es pot especificar aquesta propietat d'activació.

Bàsicament hi ha 3 escenaris d'activació dels mecanismes:

- que s'activen **sempre**. No podrem veure el contingut d'aquesta UAI si no emprem algun dels seus mecanismes d'accés
- que no s'activen **mai**. Cada vegada que accedim a aquesta UAI, aquesta mostrarà sempre el seu contingut, i a banda, podrem

emprar els mecanismes d'accés

- que s'activen per **umbral**, especificant aquest valor umbral. En aquest cas, només podrem veure directament la població d'una UAI quan hi haja menys de "valor umbral" instàncies de la classe directora de la UAI. En cas contrari, haurem d'emprar algun mecanisme d'accés

A l'exemple que s'ha presentat en seccions anterior (veure Figura 6.9 a la pàgina 134) s'ha especificat que l'accessibilitat és per *Umbral(20)*. Açò vol dir que només veuriem els llibres i el sistema no ens forçaria a emprar els mecanismes d'accés si hi han menys de 20 llibres. Amb aquest valor, l'analista hipermedial pot decidir quan una UAI té una cardinalitat de població massa elevada per a ser útil visualitzar-la directament i proporcionar-li a l'usuari mecanismes alternatius per agilitzar les búsquedes.

6.3 Gestió Avançada de Continguts

En l'actualitat, els llocs web s'han convertit en integradors d'informació heterogènea, orientats a proveir funcionalitat de diferent índole. Des d'un punt de vista de Desenvolupament per Models, les aplicacions web necessiten mecanismes conceptuals que faciliten descriure, gestionar i reutilitzar les dades i els serveis per tal de tractar amb aquesta "agregació de continguts" a un nivell d'abstracció major.

Com a resultat d'aquest requeriment, s'ha introduït una extensió a la tècnica de modelatge conceptual de manera que s'ha refinat el concepte inicial de contexte navegacional per introduir el concepte de contenidor de contingut (Unitat d'Abstració d'Informació, UAI) i que un contexte és un agregador de contenidors de contingut. Les UAI, d'altra banda, seran emprades coma a potents abstraccions encaminades a proveir de

mecanismes de reutilització que produeixen considerables beneficis al facilitar l'especificació dels contextes navegacionals.

6.3.1 Problemàtica

Els llocs web estan consolidant el seu rol en Internet. Llocs com ara Yahoo (`www.yahoo.com`), MSN (`www.msn.com`), Lycos (`www.lycos.com`) or Terra (`www.terra.es`) són els portals web més visitats . Les principals característiques d'aquestos llocs consisteixen en que proporcionen als usuaris diferents continguts d'informació (blogs, notícies, foros, ...) i serveis (correu electrònic, emmagatzemament on-line, missatgeria, ...).

La mostra la pàgina inicial del lloc de Terra. Aquesta pàgina proporciona informació de: (1) les últimes notícies del moment, (2) una llista de les novetats del dia, (3) una llista de productes en venta, (4) informació sobre activitats d'oci i (5) publicitat. Aquesta pàgina ha sigut construida a partir de l'agregació de diferents continguts o blocs. En el cas de Terra, aquestos cinc blocs composen el lloc web. Tanmateix, aquestos continguts no estan "semànticament" relacionats. És a dir, el continguts no estan directament relacionats al model conceptual subjacent.

Des d'un punt de vista metodològic, les aproximacions més rellevants en el marc de l'Enginyeria Web enfoquen els seus esforços en definir aplicacions web a partir de models conceptuals que permeten obtenir les aplicacions web a partir dels models.

Aquestes aproximacions proporcionen mecanismes que faciliten conceptualitzar i desenvolupar les aplicacions web permetent a l'analista definir els requeriments hipermedial i funcionals. Els mecanismes de modelatge hipermedial permeten: (1) definir les pàgines web com vistes en un esquema conceptual i (2) interconectar aquestes vistes per definir l'estructura navegacional de l'aplicació web. Tanmateix, considerant una pàgina web únicament com una vista sobre l'esquema conceptual, dificulta l'especificació d'aquesta agregació de continguts, on les pàgines

Figura 6.11 – Terra: Pàgina per defecte

web són construides com la composició de diferents vistes de l'esquema conceptual.

En aquesta secció es refina el concepte de contexte navegacional i es defineix un contenidor de contingut (que puguen contenir tant informació con funcionalitat), a més de definir mecanismes per la reutilització d'aquestos contenidors en la fase d'especificació navegacional del sistema. Tot açò sempre a alt nivell d'abstracció i tenint en compte que serà emprat en una estratègia de desenvolupament dirigit per models.

6.3.2 Unitats d'Abstracció de Informació, UAIs

Per tal de suportar l'agregació de continguts, els contextes navegacionals han de considerar-se com els punts d'interacció amb l'usuari que han de

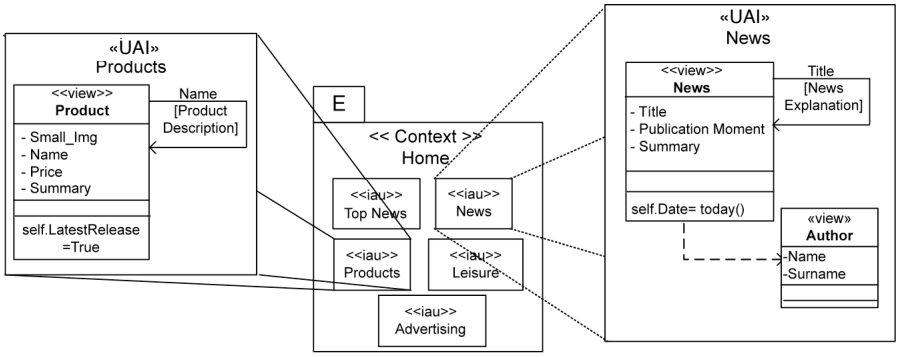


Figura 6.12 – Terra: Contexte *Home*

proporcionar accés a diferents unitats d'abstracció de contiguts (UAI). Una UAI, estereotipada amb la paraula reservada ≪UAI≫, representa una vista específica dins el diagrama de classes. Cada UAI es composa de classes navegacionals, totes relacionades entre sí per alguna relació navegacional[5]. Cada classe navegacional representa una projecció sobre una classe del diagrama de classes i cada relació navegacional implica una recuperació d'aquelles instàncies de la classe destí relacionades amb la classe orige.

La Figura 6.12 mostra el contexte *Home* del portal Terra. Aquest contexte està definit emprant cinc UAIs: "*Top News*" (Notícies Recents), "*News*" (Notícies), "*Products*" (Productes), "*Leisure*" (Esbarjo). Aquesta figura també mostra la definició de la UAI "*News*" (que proporciona les notícies del dia), i de la UAI "*Products*" (que proporciona informació sobre els últims productes en llançament).

Cada UAI ha de tenir obligatòriament una classe navegacional directora, des de la que es començarà la recuperació d'informació. A més a més, podrà tenir altres classes navegacionals complementàries, per tal de

[5] Aquest requeriment implica que sols puga mostrar-se informació relacionada dins una UAI

proporcionar informació relacionada.

En la Figura 6.12 a la pàgina anterior es pot observar que la UAI "*News*" té com a classe directora *News* i la classe complementària *Author*. A més a més, la classe *News* té definit un filtre de població *self.Date=today()* per indicar que sols es recuperen les notícies d'avuí. La UAI "*Products*" és semblant a l'anterior, amb la classe directora *Product*, sense classes complementàries i amb el filtre de població *self.latestRelease=True* per indicar que sols es recuperen aquelles notícies marcades com a d'última hora

6.3.3 Tipus d'UAIs

Es necessari distingir entre UAIs que donen suport a la navegació contextual de la que no. La *navegació contextual* és aquella navegació en la que duu informació dels objectes seleccionats des de el contexte orige [6]) al contexte destí. Aquesta informació contextual deu ser capturada per alguna UAI per tal de mostrar els detalls dels objectes seleccionats. D'aquesta maneram, podem distingir dos tipus d'UAIs:

UAIs Contextuals (representades pel símbol ©) son UAIs que instàncien la classe directora amb el/s objecte/s rebuts a través de la navegació contextual. Ha d'haver almenys una UAI contextual en: (1) contextes de seqüència i (2) en contextes d'exploració que siguen destí d'alguna relació de contexte (apareixen a l'atribut de contexte d'aquesta relació navegacional).

UAIs No Contextuals (representades pel símbol ⃠) són UAIs que no depenen de la informació contextual que puga arribar. Proporcionen sempre la població completa de la classe directora. Aquestes UAIs poden definir-se en qualsevol contexte navegacional, sense restriccions.

[6] Lligada a relacions de navegació de contexte

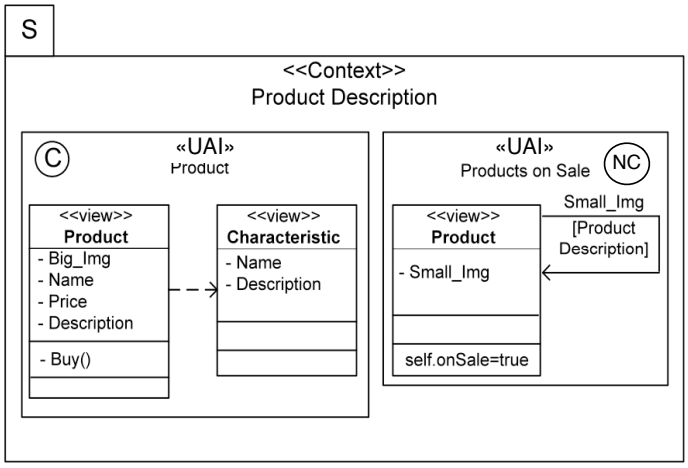


Figura 6.13 – Terra: Contexte *Product Description*

La Figura 6.13 mostra el contexte *Product Description*. Aquest contexte es composa d'una **UAI contextual** "*Product*" i una **UAI no contextual** *Products on Sale* (Productes en Venda). Quan un usuari selecciona un producte del contexte *Home* , el usuari navega) al contexte navegacional *Product Description* duent com a informació contextual el producte seleccionat. Aquest producte serà instanciat en la UAI contextual *Product* mostrant els seus atributs image, nom, preu i descripció, i les seues *Característiques* ("*Characteristics*") relacionades (veure la definició de la UAI *Producte*). Els productes en venda es mostren independentment del producte que s'haja seleccionat (ja que la UAI *Products on Sale* és del tipus no contextual).

6.3.4 Reutilització d'UAIs

Una de les tasques de la descripció d'una aplicació web, segons hem vist, és la de descriure els continguts dels nodes navegacionals. És molt habitual que en una aplicació web, diferents nodes dins el graf

navegacional, comparteixen part dels seus continguts. Per tal de facilitar la seua descripció i reutilització, es proposa construir un catàleg de continguts, de manera que, l'analista:

- empre el contingut d'aquest catàleg per crear nous nodes navegacioanls
- definisca nous contingtus
- especialitze continguts existents creant nous continguts

Cal notar que, a causa que el catàleg està format per la informació del domini del sistema, aquest catàleg sols serà aplicable a l'aplicació sobre la que es defineix i no serà doncs reutilitzable entre aplicacions.

6.3.4.1 El Catàleg de Continguts

El primer que cal decidir són els components que definiran el catàleg. Tres són les primitives que descriuen continguts i que podrien formar-ne part: (1) el contexte navegacional, (2) la unitat d'abstracció d'informació i (3) la classe navegacional.

Ens interessa que siga d'una granularitat grossa (tamany), però a la vegada que siga allò suficient menuda/modular per augmentar l'impacte en el reús (el número de vegades que és reutilitzat).

El contexte navegacional té una granularitat grossa, però com que defineix un agregat de continguts, la seua modularitat és pobra. Una UAI té una granularitat mitja, i com que defineix un contingut relacionat, la seua modularitat és elevada. Per últim, una classe navegacional té una granularitat molt baixa i la seua modularitat és extremadament alta. Per tant, s'emprarà la UAI com a component base del catàleg, pel seu compromís entre granularitat i impacte en el reús.

6.3.4.2 Especialització de Continguts Existents

Quan l'analista necessite especificar un contingut semblant a un altre que ja existisca en el catàleg, podrà extendre'l emprant el següent mecanisme d'especialització: una UAI especialitzada hereta la definició de la UAI del seu pare, i es podrà refinar per definir la informació, funcionalitat i enllaços navegacionals necessàries emprant alguna de les següents operacions de l'especialització:

- Afegir/Eliminar informació (atributs navegacionals)
- Afegir/Eliminar accés a funcionalitat (operacions navegacionals)
- Afegir/Eliminar classes navegacionals complementàries (i la seua relació navegacional associada)
- Canviar el tipus d'una relació navegacional
- Afegir/Eliminar/Redefinir filtres de població a les classes navegacionals

L'especialització d'UAIs es realitza per mig d'un operador is_a que acompleix açò descrit. La Figura 6.14 a la pàgina següent mostra la definició de la UAI *Top News*. Aquesta UAI ha sigut especialitzada de la UAI *News*. En aquest cas, la UAI pare (*News*) ha sigut refinada afegint un atribut nou (*Image*) i redefinint el seu filtre de població (per tal de recuperar sols les notícies d'última hora).

6.3.5 Conclusions

En l'actualitat, certes polítiques de màrketing, captació d'usuaris, ..., han fet que molts llocs web s'hagen convertit en aplicacions que proporcionen gran quantitat d'informació no cohesionada ni relacionada, però barrejada. A causa de l'interés en especificar i desenvolupar aplicacions

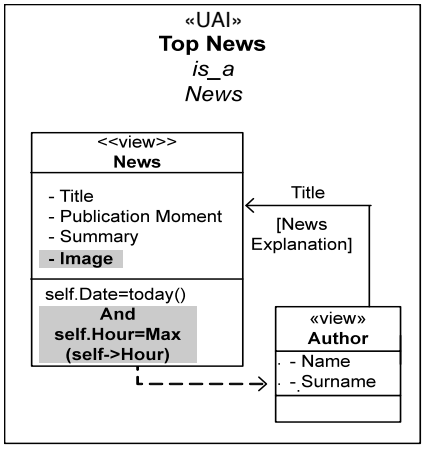


Figura 6.14 – Terra: UAI especialitzada *Top News*

web amb aquestos requeriments, s'ha proposat una solució de d'un punt de vista del modelatge conceptual per donar suport metodològic al seu desenvolupament.

Aquesta solució refina algunes primitives de modelatge proposades pel mètode OOWS, en concret els *contextes navegacionals*, introduint el concepte d'**Unitat d'Abstracció d'Informació** ó UAI. Aquestes UAIs permeten representar en la fase de modelatge conceptual, els diferents continguts heterogenis que poden definir una pàgina web.

Aquesta aproximació redefineix el concepte de contexte navegacional de manera que es converteix en un agrupador de UAIs. Aprofitant aquesta estratègia de definició de blocs de continguts (UAI), s'ha aprofitat per introduir mecanismes enfocats a la reutilització d'aquestos blocs en diferents contextes, facilitant per tant la construcció dels models navegacionals, inclús aplicant mecanismes d'especialització entre aquestos.

6.4 Personalització d'Aplicacions Web

La demanda per aplicacions web que tenen en compte les diferents necessitats i inquietuts dels usuaris ha anat creixent aquestos darrers anys. La personalització involucra un ample rang de tecnologies, aproximacions, eines, etc. per millorar l'experiència d'usuari. Aquesta secció descriu l'ús de tècniques de modelatge conceptual en el desenvolupament de programari per al disseny d'aplicacions web personalitzades.

6.4.1 Introducció

Amb la consolidació de la "*World Wide Web* " (WWW) com a plataforma de desenvolupament. Aquesta es caracteritza per la necessitat de suportar ràpids canvis tecnològics i per la de definir aplicacions personalitzades i adaptades per a uns requeriments específics d'usuaris. L'*Adaptació* és el mecanisme per definir les regles que permeten a les aplicacions (web) donar un suport adequat als perfils d'usuari, històrics de navegació, dispositius, conexions de xarxa, ... Les applicacions de comerç electrònic són exemples representatius de sistemes on l'adaptació a nous entorns i requeriments és un factor crític. En aquest contexte, els entorns de producció automàtic de programari poden suposar un avanç significatiu per accelerar el procés de desenvolupament i augmentar les capacitats de reusabilitat. Tot açò va unit al fet que cada dia és més habitual i necessari fer adaptacions dinàmiques a les aplicacions atenent a les preferències dels usuaris.

Avuí en dia, les aproximacions que tracten d'alguna manera la personalització varien significativament [2]: des d'estratègies que simplement generen pàgines fins sistemes complexes de predicció basats en contigunt, patrons de reconeixements de comportament d'usuaris, algorismes d'aprenentatge i mineria de dades. Moltes d'aquestes aproximacions consideren la personalització des d'un punt de vista d'implmentació.

D'acord amb [57], les tècniques per recollir la informació de l'usuari són la de l'extracció explícita de perfils d'usuari, l'inferència implícita de perfils d'usuari i dades legades. Les tècniques més emprades per analitzar aquesta informació estan basades en regles de filtrat.

Tanmateix, nosaltres considerem l'ús de tècniques de modelatge conceptual per a un correcte tractament de la inherent complexitat de les aplicacions web. En [62] i [67], els aspectes d'adaptació en aplicacions web són discutits enfocant la necessitat de mètodes de modelatge per desenvolupar aplicacions web adaptables. El mètode WSDM [37] és un mètode dirigit per l'audiència que defineix els objectes d'informació tenint en compte els requeriments d'informació dels usuaris finals com punt clau i central per desenvolupar les aplicacions web. En [106] els autors defineixen diferents escenaris d'acord a perfils d'usuari o preferencies. En [15] es proposa un llenguatge per dissenyar aplicacions web, tenint en compte un model de personalització on els usuaris i grups poden ser explícitament especificats en l'estructura conceptual del sistema d'informació.

La proposta que presenta aquesta tesis proporciona una contribució en aquest contexte dirigint l'esforç en proporcionar a nivell de modelatge conceptual conceptes que permeten representar requeriments de personalització. En aquesta línia, s'ha obert un camí principal d'enriquiment dels models navegacionals [94, 89], que posteriorment ha sigut extés [102, 104] per dotar de major expressivitat en l'adaptativitat dels sistemes.

6.4.2 Disseny d'Aplicacions Web Personalitzades

En la fase de modelatge conceptual de les aplicacions web [62, 106], l'adaptació pot dur-se a tres nivells:

- el nivell de *contingut* d'informació, que defineix quina informació està disponible per un usuari concret

- el nivell de *presentació*, que defineix la manera en què aquesta informació es proporciona
- el nivell d'estructura de *navegació*, que defineix la manera en que s'organitza l'accessibilitat pel sistema

L'adaptativitat per a cadascun d'aquestos nivells pot ser vista de des de dos **aspectes** diferents: *estàtic* i *dinàmic*. L'adaptativitat estàtica es defineix per complet a nivell d'anàlisi (disseny). L'adaptació dinàmica depèn de condicions en temps d'execució.

A nivell de **granularitat**, es poden considerar dos nivells a l'hora d'aplicar adaptativitat: polítiques de personalització sobre *grups d'usuari* o sobre *usuaris individuals*.

D'altra banda, l'adaptativitat pot tenir en compte l'**entorn**, canviant o adpatant el sistema depenent dels recuros destí (dispositiu, localització, conexió de xarxa, etc.).

Amb aquestos conceptes, hi ha una terminologia acceptada en la comunitat de la Hipermèdia Adaptativa [21] que defineix diferents tipus de sistemes:

- **Adaptatius**. Sistemes que deprenen en base a unes regles d'adaptativitat, per tal d'adaptar-se dinàmicament a les condicions d'ús del sistema.
- **Adaptables**. Sistemes que solen emprar un framework d'adaptació preparat per "recodificar o introduir" més requeriments d'adaptativitat.
- **Adaptats**. Sistemes que en la seua construcció han tingut en compte regles d'adaptativitat i aquestes han sigut implementades per donar suport a l'especificació. No obstant, el sistema no està preparat per "recodificar o introduir" nous requeriments d'adaptativitat en temps d'execució.

- **No Adaptades**. Sistemes en els que no s'ha tingut en compte en cap moment requeriments d'adaptativitat.

Un escenari concret d'adaptativitat i amb un especial interés és l'adaptativitat a l'usuari, que sol anomenar-se *personalització*. D'aquesta manera, i segons la classificació anterior, hi haurà aplicacions "personalitzatives", personalitzables, personalitzades i no personalitzades.

Atenent a aquesta classificació, les primitives d'abstracció d'OOWS permeten únicament la *personalització estàtica als nivells de contingut, presentació i navegació.* Respecte a la *personalització dinàmica, sols permet als nivells de contingut i navegació.* També dóna suport *tant a la granularitat de grup com a la individual.* Les següents seccions tractaran de justificar com es dóna suport tant a la personalització estàtica com dinàmica.

6.4.3 Personalització Estàtica en OOWS

La personalització estàtica del nivell de navegació s'obté enllaçant un mapa navegacional amb un tipus d'usuari, predefinint la seua estructura de navegació en temps d'anàlisi. Amb els contextes navegacionals per a cada usauri podem personalitzar el contingut mitjançant les classes navegacionals (atributs i operacions), relacions navegacionals i filtres de població. Els patrons de presentació especificats per als contextes navegacionals defineixen la personalització estàtica a nivell de presentació.

La personalització definida a la granularitat de grup s'aconsegueix pel fet que cada mapa navegacional està associat a un tipus d'usuari. D'aquesta manera, els usuaris que pertanyen a aquest tipus comparteixen les mateixes propietats. Tanmateix, no es pot definir personalització individual estàtica, ja que açò "força" a conèixer les instàncies d'usuaris que hi haurà al sistema, quan encara estem en fase de modelatge.

6.4.4 Personalització Dinàmica en OOWS

La personalització dinàmica permet la definició de regles i condicions a nivell conceptual que deuen ser avaluades en temps d'execució depenent de l'usuari que estiga interactuant amb el sistema.

El mètode OOWS proveeix de la primitiva Usuari Actiu per descriure la personalització dinàmica a nivell de contingut. Aquesta personalització dinàmica s'aconsegueix emprant el concepte usuari actiu per definir de filtres de població associat a les classes navegacionals, condicionant la recuperació d'informació en funció del valor d'alguna de les propietats de l'usuari actiu.

La personalització dinàmica a nivell de navegació implica la possibilitat de redefinir el mapa navegacional en temps d'execució. Les implicacions d'aquest nivell de personalització són considerables i poc habituals. Sols s'ha considerat treballar en aquesta línia en la possibilitat que l'aplicació web puga adequar les opcions de navegació (creant subconjunts o ordenant els enllaços) atenent al mapa navegacional predefinit. En [104] s'estudien polítiques de personalització dinàmica a la navegació aplicades a aquest mètode.

6.4.5 Conclusions

En aquesta secció s'ha presentat com el mètode OOWS dóna suport al disseny d'aplicacions web personalitzades. En aquesta proposta s'argumenta que també es poden dissenyar i realitzar aplicacions web que inclouen certs aspectes de personalització, d'una manera més extensible i mantenible mitjançant la seua definició mitjançant models conceptuals.

En OOWS els usuaris són una part principal a l'hora d'especificar un sistema. D'aquesta manera, el sistema és dissenyat de manera personalitzada per a cada tipus de responsabilitat (representada per cada tipus d'usuari).

Com es pot veure en les darreres conferències sobre *Hipermèdia*

Adaptativa [144, 4], l'ús de models conceptuals apareix com una manera natural de descriure els requeriments d'adaptativitat. El treball que es presenta en aquesta secció defineix les línies fonamentals de treball en aquesta àrea. Un treball més extens que s'ha construït sobre aquest es pot veure en [101].

6.5 Herència de Propietats Navegacionals

Avuí en dia, les aplicacions web han evolucionat en sistemes més complexes, tenint un alcanç a gran nombre d'usuaris. Si considerem la navegació com una part emergent de les característiques de les aplicacions web, és necessari tractar d'una manera adequada la manera en que es proporcionen els mecanismes que permeten reutilitzar les especificacions navegacionals. A continuació es presenta un mecanisme per capturar adequadament la semàntica navegacional per poder emprar l'**especialització navegacional** com a eina per extendre i reutilitzar les característiques i requeriments navegacionals.

6.5.1 Motivació

La web s'ha convertit en l'interfície més emprada per compartir dades en la Xarxa. Actualment, es pot considerar com la plataforma estàndar per a un ample rang d'aplicacions. A més a més, cada vegada més companyies tenen la necessitat de poder adaptar i personalitzar els seus continguts atenent a les preferències dels usuaris [8]. Apareix la necessitat de donar un correcte suport per al tractament dels potencials usuaris del sistema. Aquestos fets impliquen un increment en la complexitat dels sistemes i la incorporació de requeriments de sistema directament expressats en forma d'objectius d'usuaris.

Des d'un punt de vista de Modelatge Conceptual, s'han de proveir mecanismes per estructurar l'accessibilitat dels usuaris al sistema i poder

descriure com aquestos tenen accés a la informació i la funcionalitat. Els mètodes més rellevants que segueixen els principis de l'Enginyeria Web [108, 15, 36, 65, 55] ho fan d'una manera o una altra. Tots aquestos fan especial menció als requeriments de navegació i presentació.

El tractament de la navegació s'aconsegueix introduint un model navegacional. Aquest model navegacional sol definir-se mitjançant grafs (nodes i enllaços) relacionant-los completa o parcialment amb els usuaris. D'aquesta manera es pot representar d'una manera senzilla la vista d'un usuari, com el conjunt de nodes i enllaços que pot explorar.

La intenció es proporcionar expressivitat per millorar la reutilització, introduint el concepte d'**especialització navegacional** en Models Conceptuals, adaptant el concepte convencional d'especialitació, aplicant-lo als entorns de modelatge d'aplicacions web i estudiant el seu impacte a l'aplicar-lo a la especialització entre usuaris.

6.5.2 Problemàtica

El model navegacional d'un mètode de producció d'aplicacions web serveix per representar els requeriments navegacionals [110, 13]. Aquest model descriu els camins d'un usuari per aconseguir el seu objectiu en el sistema: accedir a la informació i la funcionalitat a la que l'usuari ha de necessàriament tenir accés. És per tant, aquesta associació entre accessibilitat a la informació i funcionalitat la que ha de guiar la construcció d'aquestos models navegacionals.

Des d'aquest punt de vista, existeixen bàsicmanet dos aproximacions per construir els models navegacionals:

1. Moltes aproximacions consideren únicament un **únic graf navegacional** on s'exposen els requeriments d'accés a la informació i funcionalitat de manera global. Després, els usuaris són enllaçats/relacionats amb aquestos nodes (i enllaços), representant

l'accessibilitat concreta a la informació i funcionalitat per a cada tipus d'usuari.

2. Altres aproximacions fan un estudi previ dels tipus d'usuari i quina hauria de ser la seua accessibilitat concreta a la informació i funcionalitat. Després, el model de navegació es crea a partir d'aquest estudi, normalment associant **un graf navegacional independent per a cada usuari**.

La reutilització de peces del model navegacional apareix quan diferents usuaris comparteixen accés a la mateixa informació i funcionalitat. Però, no hi ha una solució trivial. Si veiem els principals avantatges i inconvenients de cada grup d'aproximacions:

1. En les aproximacions on s'empra un únic graf navegacional, l'accessibilitat d'un usuari s'obté a partir de les relacions de visibilitat definides sobre els nodes i els enllaços. En aquestes aproximacions, un node pot tenir varios usuaris associats, permetent que tots tinguen accés al mateix contingut. D'alguna manera, s'està reutilitzant aquest contingut per a tots els usuaris. Tanmateix, no es fàcil representar la compartició parcial d'un (implica etiquetar parcialment el contingut) per diferents usuaris. En aquest cas, el node ha de ser dividit en diferents nodes: un amb les parts comunes i compartides per tots els usuaris i altres amb les parts específiques per a cada usuari. És per açò que el graf navegacional descrit no és totalment independent de l'usuari i la seua accessibilitat a la informació i funcionalitat. Aquest fet dificulta el "manteniment" d'un node, ja que els canvis afecten a més d'un tipus d'usuari. A més a més, aquest tipus d'aproximacions no poden tenir en compte d'una manera sencilla aspectes de personalització, ja que els nodes navegacionals no estan concebuts per a un tipus únic d'usuari.

Segons s'ha discutit a la Secció 6.4, la personalització és l'adaptació al tipus d'usuari.

2. Al segon grup d'aproximacions, es construeix un graf navegacional per a cada usuari del sistema. Aquest procés de construcció és guiat pels requeriments d'accessibilitat de cada tipus d'usuari a un conjunt d'informació i funcionalitat. Aquestes aproximacions permeten abstraure la definició de les característiques navegacionals per a cada usuari, de manera independent als altres. Quan apareix una modificació en els requeriments d'accés d'algun dels usuaris, els canvis al seu Mapa Navegacional només l'afectaran a ell. D'altra banda, quan usuaris diferents comparteixen accés a informació i funcionalitat, aquestes aproximacions forcen a la definició de nodes repetits en diferents grafs (un en cada mapa navegacional de cada usuari). D'aquesta manera, s'introdueix redundància al models. Malgrat açò, aquesta aproximació afavoreix la personalització ja que cada graf pertany a un tipus d'usuari concret i sobre aquest es poden aplicar de manera controlada les polítiques de personalització.

Es per tot açò que s'ha triat la segona de les aproximacions per abordar el procés de definició navegacional i sobre la que es definirà el concepte d'*herència navegacional*, aplicant-ho al model navegacional definit pel mètode d'OOWS. D'aquesta manera s'aconsegueix l'objectiu de poder especificar els requeriments d'una manera abstracta per a cada tipus d'usuari, i després definir relacions entre usuaris per reutilitzar aquelles parts comunes entre usuaris.

6.5.3 Herència en Models Navegacionals

Des d'un punt de vista convencional, l'herència de classes en un Diagrama de Classes és un mecanisme per compartir propietats i comportament

entre classes, establint un mecanisme clar per a la reutilització d'especificacions. En OO-Method, aquestes propietats heretades són: atributs, operacions, relacions entre classes, restriccions d'integritat, etc. Però també ho són altres relacionades amb el comportament, com ara les seqüències de vida vàlides dels objectes definides al model dinàmic, o els canvis d'estat produïts per les operacions definides al model funcional. Tradicionalment, els operadors d'especialització *es-un* s'han definit doncs a nivell estructural i de comportament dels objectes.

En aquest punt, ens podem plantejar qué passaria si aplicarem aquest mateix concepte a classes que representen els usuaris del sistema. Des d'un punt de vista tradicional, allò que aconseguiriem és que compartiren estructures d'informació i comportament, però res més. Ara bé, si extenem les propietats dels usuaris i els podem associar mapes navegacionals (com s'ha vist al Capítol 4), podriem considerar que aquestes propietats també s'hereten. És a dir, si definim una relació d'herència *es-un* entre usuaris, l'usuari especialitzat hereta l'accés a l'estructura navegacional de l'usuari pare, tenint accés als nodes i enllaços definits al mapa del pare. A aquest concepte l'anomenarem d'ara endavant **especialització navegacional**.

Però, com es podria aplicar aquesta especialització navegacional atenent a l'aproximació de descripció navegacional que emprem?

- En les aproximacions que descriuen un únic graf navegacioanl, seria suficient en associar tots els nodes als que el pare té accés a l'usuari especialitzat. Açò conduiria a la mateixa accessibilitat per a l'usuari especialitzat.

- En les aproximacions que descriuen un graf per a cada usuari del sistema, es podria considerar aquest graf com una propietat adicional (emergent) de l'usuari. D'aquesta manera, quan un usuari s'especialitza d'un altre usuari, aquest graf també s'hereta,

permetent a l'usuari especialitzat accedir i navegar els mateixos continguts que l'usuari pare.

Com es pot observar, aquesta definició d'especialització navegacional pot ser emprat com una eina per estructurar i reutilitzar l'espai navegacional definits pels usuaris de les aplicacions web. Defineix un mecanisme d'herència addicional, que s'aplicarà només a classes que representen usuaris, i que, de la mateixa manera que esdevé amb l'herència estructural i de comportament, permetrà:

- l'herència
- la redefinició
- la cancel·lació
- l'addició

de propietats navegacionals als grafs navegacionals dels usuaris especialitzats.

L'*herència* permet obtenir les mateixes propietats navegacionals de l'usuari base (**compatibilitat navegacional**). És el comportament per defecte al definir una relació *es-un* entre usuaris. La **redefinició** permet modificar alguna propietat navegacional, definint una estructura navegacional més concreta per a l'usuari especialitzat. La **cancel·lació** anul·la l'herència d'alguna/es propietat/s navegacional concreta. Per últim, l'**addició** permet especificar més propietats navegacionals que sols són accessibles per a l'usuari especialitzat.

Finalment, si tenim en compte aquestos requeriments que ha de complir aquesta especialització navegacional, pareix clar que és més interessant aplicar-ho a aproximacions en les que existeix un graf navegacional per a cada usuari. D'aquesta manera, les modificacions d'aquestos grafs per l'aplicació de qualsevol d'aquestes operacions de l'especialització

navegacional estan més controlades i permeten un manteniment més clar i precís que l'altre grup d'aproximacions.

En la següent secció es veurà com s'apliquen aquestos conceptes al mètode OOWS.

6.5.4 Especialització Navegacional en OOWS

OOWS defineix la navegació emprant l'aproximació "un graf per cada usuari". A més a més, permet especialitzar els usuaris, considerant així l'herència de l'accés a la informació i la funcionalitat. Per tal d'explicar amb precisió les capacitats expressives d'aquest mecanisme d'especialització navegacional, s'empra la taxonomia de relacions d'herència definida en [97]:

- *Interpretació Semàntica*: significat de les abstraccions
  - **Especialització**. La interpretació del mecanisme d'herència proposat defineix que: "un mapa navegacional pot ser especialitzat en un altre, de manera que el mapa especialitzat captura requeriments navegacionals més específics que el seu mapa base".
- *Relació entre Propietats*: Herència i Visibilitat de Propietats
  - **Herència de Propietats**: l'especialització proposta permet heretar totes les propietats d'un mapa navegacional i les seues propietats (conjunt de contextes navegacionals, subistemes navegacionals i enllaços navegacionals).
    - **Propietats Heretades**: sols és possible heretar un mapa sencer d'un usuari, del que s'heretaran totes les propietats (contextes, subsistemes i enllaços).
    - **Propietats Modificades**: un mapa navegacional es veu modificat al modificar la definició dels seus components.

Així, si modifiquem o redefinim un contexte navegacional, aquest es converteix en un contexte navegacional especialitzat per a aquest usuari. Les operacions de modificació aplicables poden veure's en les Seccions 6.5.5 i 6.5.6.

  * **Noves Propietats**: es poden introduir nous contextes i subsistemes navegacionals a un mapa, definint així noves relacions navegacional. Aquestos nodes, probablement definiran nous enllaços navegacionals.
  * **Propietats Cancel·lades**: es pot cancel·lar l'herència tant contextes, com subsistemes, com enllaços navegacionals. Açò provoca que aquestos no siguen accessibles al mapa especialitzat. Quan un contexte es cancel·la en herència, tots els enllaços navegacionals relacionats amb ell també es cancel·len. Quan un subsistema es cancel·la, tots els nodes (contextes i subsistemes) que conté, també són cancel·lats en herència.

- **Visibilitat de Propietats**: la visibilitat és *ascendent*. Sols el mapa navegacional especialitzat pot obtenir informació sobre el seu mapa navegacional base.

Aquestes propietats proporcionen una caracterització de l'especialització aplicat a la definició d'aspectes navegacionals en OOWS, permetent el reús de mapes navegacionals. Un mapa navegacional constitueix el principal constructor per especificar navegació en OOWS.

A continuació s'enumeren les operacions de modificació de mapes especialitzats en OOWS. Aquestes operacions s'apliquen a les propietats dels mapes navegacionals, és a dir, als contextes navegacionals, subsistemes navegacionals i enllaços navegacionals. A causa que els enllaços navegacionals són realment informació derivada a partir de l'especificació dels nodes, a continuació es presenten aquestes operacions agrupades per la seua aplicació a contextes i subsistemes navegacionals.

6.5.5 Operacions d'Especialització de Contextes

Les operacions que es poden aplicar per modificar un contexte navegacional especialitzat són les següents:

- convertir un contexte de seqüència en un contexte d'exploració. Açò provocarà la definició d'un enllaç d'exploració al mapa amb destí aquest contexte
- convertir un contexte d'exploració en un contexte de seqüència. Açò eliminarà l'enllaç d'exploració associat. A més, el contexte (ara de seqüència) haurà de ser accessible a través d'algun altre contexte (per mig d'un enllaç de seqüència)
- modificar el tipus d'una Unitat d'Abstracció d'Informació (UAI) dins el contexte. Si la UAI és Contextual, pot passar-se a No-Contextual, o a l'inrevés. Només hi ha la restricció que si el contexte de navegació és de tipus Seqüència, almenys ha d'existir una UAI de tipus Contextual
- afegir/eliminar una UAI dins el contexte. Tot contexte navegacional ha de tenir almenys una UAI. I si el contexte navegacional és de Seqüència, almenys una UAI Contextual
- modificar el contingut d'una UAI dins el contexte. Per a fer-ho, es poden aplicar les següents operacions:
  - afegir/eliminar una classe navegacional complementària[7]
  - modificar una classe navegacional, afegint/llevant atributs navegacionals o operacions navegacionals, o bé afegint/modificant/esborrant un filtre de població associat a la classe

[7]No es permet afegir o eliminar la classe directora.

  - afegir/eliminar una relació de dependència de contexte (no defineix ni esborra cap capacitat de navegació)
  - modificar una relació de dependència de contexte, convertint-la en una relació de contexte. Açò causarà la definició d'un enllaç navegacional al mapa
  - afegir/eliminar una relació contextual entre classes. Açò durà a la creació/eliminació d'un enllaç al mapa navegacional
  - modificar una relació de contexte canviant algun dels seus atributs: atribut de rol, atribut d'enllaç o atribut de contexte. Canviar l'atribut de contexte força al canvi del destí de l'enllaç navegacional associat
  - modificar una relació de contexte convertint-la en una relació de dependència de contexte. Caldrà eliminar l'enllaç navegacional associat la relació de contexte
  - afegir/modificar/esborrar qualsevol dels patrons de presentació associats

6.5.6 Operacions d'Especialització de Subsistemes

Les operacions que es poden aplicar per modificar un subsistema navegacional especialitzat són les següents

- convertir un subsistema de seqüència en un subsistema d'exploració. Açò provocarà la definició d'un enllaç d'exploració al mapa amb destí aquest subsistema
- convertir un subsistema d'exploració en un subsistema de seqüència. Açò eliminarà l'enllaç d'exploració associat. A més, el subsistema (ara de seqüència) haurà de ser accessible a través d'algun altre contexte (per mig d'un enllaç de seqüència)
- canviar el node definit com "Per Defecte" ó "*Home*"

- afegir/eliminar un contexte navegacional
- afegir/eliminar un subsistema navegacional

6.5.7 Example: Parts de Treball

Aquesta secció mostra un cas d'estudi desenvolupat on s'han aplicat els principis de l'especialització navegacional. Aquest cas d'estudi tracta d'un escenari real basat en una companyia que necessitava crear una aplicació web per gestionar les tasques realitzades pels seus treballadors tècnics.

Cada *tècnic* ha d'introduir al sistema les tasques que ha realitzat dins el contexte d'un projecte. Cada projecte pertany a un tipus de projecte. Aquestes tasques són d'un tipus predefinit de tasques (documentació, programació, anàlisi, etc.). Cada tècni ha de gestionar únicament (crear, modificar i esborrar) sols les seues tasques. A més a més, un tècni ha de poder veure els projectes als que participa (realitzant tasques).

Els *responsables* són un tipus especial de tècnics que són responsables d'almenys un departament. Tenen control sobre les tasques, tècnics i projectes assignats al seu departament.

Per últim, existeixen els *directors* que són responsables de crear nous projectes, tipus de projectes i departaments dins la companyia. Són responsables de promocionar un tècnic a responsable i un resposable a director. També ho són de degradar un responsable a tècnic.

El sistema fou modelat emprant l'aproximació OO-Method/OOWS. En la primera etapa, s'especificaren els aspectes estructurals i dinàmics del sistema mitjançant els Diagrama de Classes, Model Dinàmic i Model Funcional d'OO-Method. En una segona fase, es construí el Model Navegacional d'OOWS, tenint en compte els requeriments de l'aplicació Web.

6.5.7.1 Modelatge Conceptual amb OO-Method

La Figura 6.15 a la pàgina següent mostra el diagrama de classes descrit amb OO-Method per aquest sistema d'informació. En aquest model, els technician s (tècnics), responsible s (responsables) i director s són considerats agents[8] del sistema. Aquestes classes tenen relacions d'agent definides (línies discontínues) per representar aquella informació i funcionalitat que poden emprar. A més a més, aquestes classes creen una jerarquia d'especialització, de manera que un responsable és considerat una especialització d'un tècnic i un director d'un responsable. D'aquesta manera s'especifica que un usuari especialitzat extén l'accessibilitat del seu usuari pare.

A més de la descripció estructural, es feu la descripció funcional. Com que aquesta no és necessària per explicar l'especialització navegacional, no s'entrarà en més detalls.

6.5.7.2 Model Navegacional

Una vegada descrita la part estructural, cal descriure la part navegacional. Per a cada usuari s'ha de definir el seu mapa navegacional. A continuació es mostren aquests mapes per a cada usuari, i pot observar les implicacions de l'herència navegacional d'usuaris.

El mapa navegacional d'un technician, té dos contextes (veure Figura 6.16 a la pàgina 164): un per gestionar les seues tasques (*context tasks*) i altre per veure els projectes en que participa (*context project_types*).

Un responsible és un tècnic amb accés addicional a informació i funcionalitat sobre projectes i tècnics. Com que és un usuari especialitzat d'un tècnic, heretarà el seu mapa i podrà definir aquests requeriments addicionals. Per aconseguir-ho, hi ha dos contextes nous: el contexte *projects* i el contexte *technicians*. La resta de contextes del mapa són

[8] *Agent* és el terme que empra OO-Method per descriure un usuari.

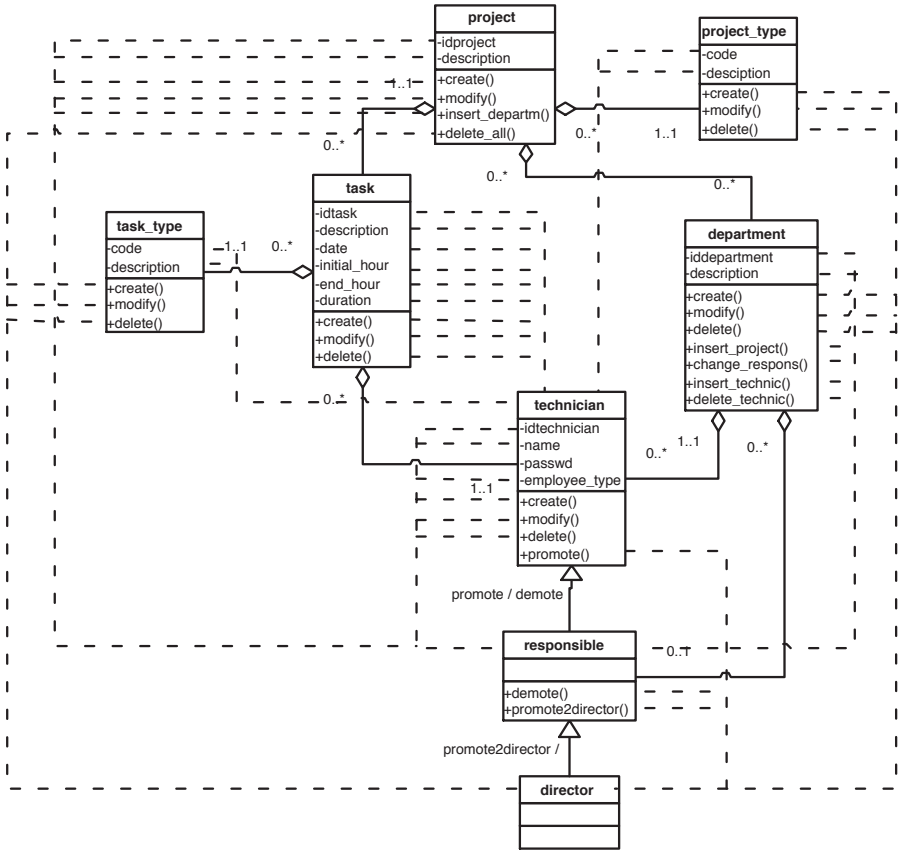


Figura 6.15 – Parts de Treball: Diagrama de Classes

heretats, i es pot veure amb el prefixe al nom del contexte (empra l'usuari propietari del mapa on està definit el contexte).

Un responsible deu crear i modificar projectes. Per representar aquest requeriment, el sistema li deu proporcionar informació sobre els seus projectes (idenficador de projecte i descripció), tipus de projectes (codi i descripció) i els tècnics que treballen en cada projecte (identificador i nom). El contexte *projects* representa aquest requeriment i pot veure's

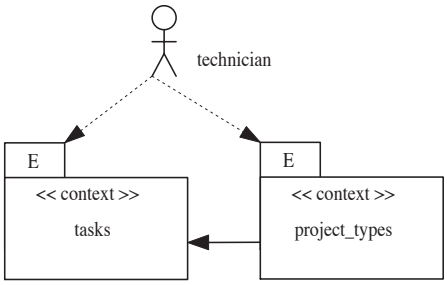


Figura 6.16 – Parts de Treball: Mapa del Tècnic

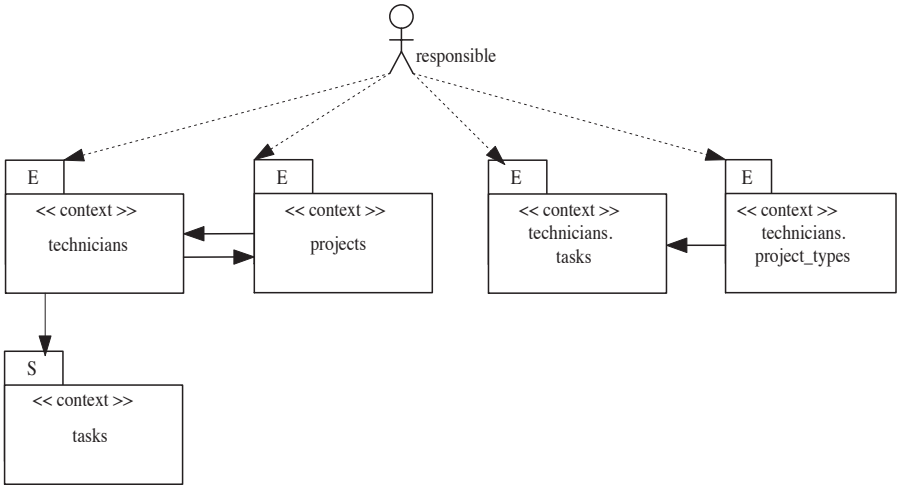


Figura 6.17 – Parts de Treball: Mapa del Responsable

a la Figura 6.18 a la pàgina següent.

De la mateixa manera, un director director heretarà el mapa dels responsables, afegint nous contextes per poder gestionar tipus de projectes, tipus de tasques, departaments, responsables i tècnics (hi ha un contexte nou per cadascun d'aquestos requeriments). Però, un requeriment especial s'afegí: a causa de l'estàtus dels directors, el sistema no ha de demanar-los que introduisquen les tasques. Per aconseguir-ho, cal

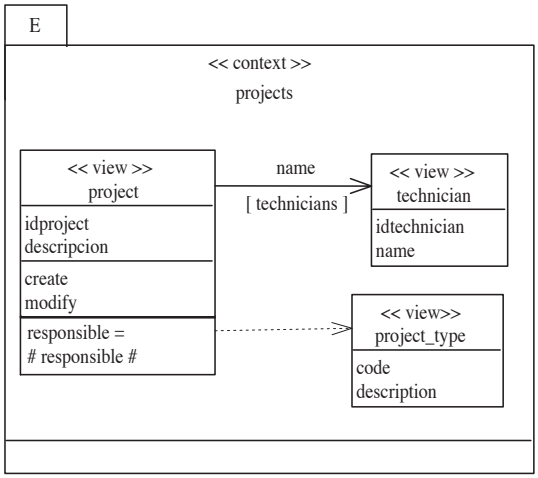


Figura 6.18 – Parts de Treball: Contexte *Projects* del Responsble

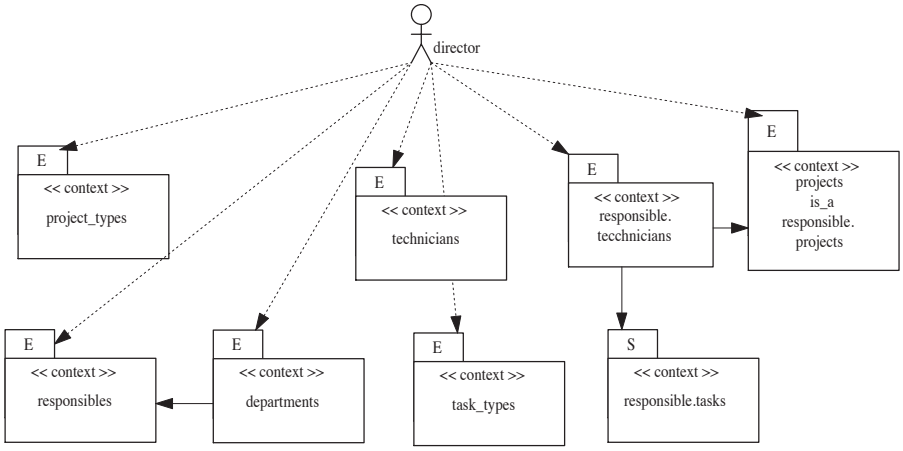


Figura 6.19 – Parts de Treball: Mapa del Director

cancel·lar l'herència dels contextes technician.*tasks* i technician.*projects*. La Figura 6.19 mostra el mapa del director amb aquestos canvis.

Un director pot realitzar realitzar més tasques sobre el sistema que

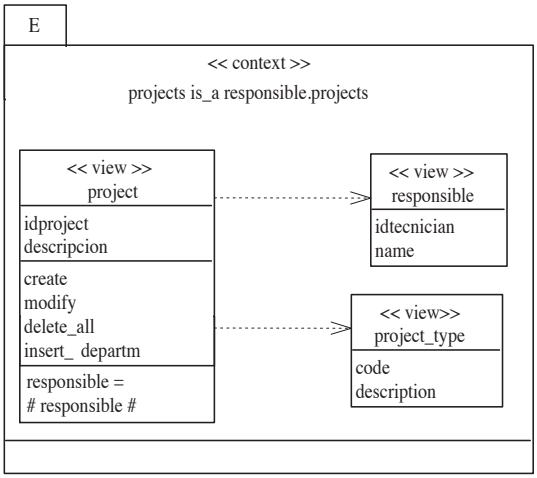


Figura 6.20 – Parts de Treball: Contexte Projectes del Director

un responsable per gestionar un projecte (veure les relacions d'agent al diagrama de classes). Un director opt asignar un projecte a un departament, o esborrar un projecte i totes les seues tasques. A més a més, pot obtenir informació sobre cada projecte i el seu responsable, sense veure els tècnics que hi treballen.

És per açò que es pot especialitzar i reutilitzar el contexte homòleg *projects* d'un responsable i modificar-lo, esborrant les classes complementàries task s i technician s i afegint dues noves classes complementàries: responsible (per veure el responsable del projecte) i project_type (per veure el tipus de projecte). El contexte final després d'aplicar aquestes modificacions es pot veure a la Figura 6.20.

Al mapa navegacional del director, s'ha emprat la següent expressivitat amb relació a l'herència navegacional:

- Herència de propietats navegacionals dels responsables (contextes responsible.*technicians* and responsible.*tasks*) and technician (technician.*tasks* and technicians.*project_types*.

- Introducció de noves propietats navegacionals: afegir contextes d'exploració *project_types*, *technicians*, *responsibles*, *departments* i *task_types*.
- Cancel·lació de propietats: esborrat dels contextes technicians.*tasks* i technicians.*project_types*
- Modificació de propietats: el contexte *projects* ha sigut modificat mitjançant especialització a partir del seu contexte pare *responsible.projects*, afegint i llevant classes complementàries.

6.5.8 Conclusions

La contribució d'aquesta secció ressenya la importància de modelar i gestionar els usuaris a nivell conceptual a l'hora d'especificar les extensions navegacionals als models conceptuals.

Es presenta l'**especialització d'usuaris** als models conceptuals, especialment al models navegacionals, indicant els seus efectes. Aquesta especialització implica l'herència (compartició) d'accés al nodes de contingut i per tant a la *informació i funcionalitat* del sistema d'un usuari especialitzat en base al seu usuari "pare".

Així, s'introdueix el reús de "peces" dels models navegacionals dins entorns Orientats a Objectes, definint expressivitat per adaptar-los a nous requeriments de l'usuari especialitzat.

En aquest sentit, s'ha definit un framework en OOWS per especialitzar models navegacionals considerant aquestos requeriments d'especialització d'usuaris, amb la possibilitat d'heretar, afegir, modificar i cancel·lar propietats navegacionals.

Capítol 7

Model de Presentació

7.1 Introducció

Amb el model navegacional s'ha descrit la navegabilitat dels usuaris pel sistema, definint el seu mapa navegacional que organitza els nodes de continguts (contextes navegacionals) i les seues relacions entre sí mitjançant enllaços navegacionals (relacions de contexte). D'aquesta manera, un mapa navegacional descriu una vista parcial del sistema per a cada usuari.

Una vegada definit aquest model navegacional, cal descriure el requeriments de presentació de les aplicacions web emprant el **Model de Presentació**. Aquest model es construix sobre el model navegacional anotant-lo amb informació sobre com s'haurà de presentar i permet introduir certs aspectes relacionats amb l'aparença de les aplicacions.

Aquestos requeriments de presentació són definits mitjançant patrons que s'asocien a les primitives navegacionals, en l'àmbit dels contextes navegacionals: classes navegacionals, enllaços navegacionals, mecanismes de búsqueda, etc.

Es pot veure aquest model de presentació com una extensió del model de navegació que s'enriqueix amb un conjunt de patrons bàsics de

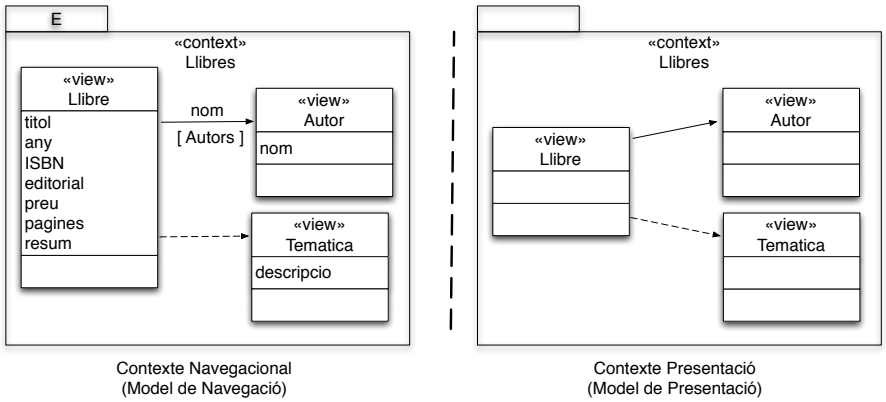


Figura 7.1 – Contexte de Presentació

presentació. Per tal d'especificar les propietats de presentació es s'utilitza únicament l'estructura del contexte.

Com es pot observar en la Figura 7.1 a la part dreta, s'ha emprat l'estructura del contexte navegacional del model de navegació per descriure les propietats de presentació. A continuació es descriuen el patrons de presentació i la seua aplicabilitat a les primitives del model de navegació.

7.2 Patrons del Model de Presentació

A continuació es presenten els patrons de presentació bàsics que el mètode defineix. Com s'ha comentat abans, l'objectiu d'aquestos patrons és descriure a alt nivell d'abstracció, les característiques de presentació dels elements conceptuals més adients per a entorns web.

A mode aclaratori, va a emprar-se un exemple per tal de descriure els patrons de presentació. La Figura 7.2 a la pàgina següent mostra el mapa navegacional i el contexte *Llibres* de la Biblioteca.

Com es pot observar, el contexte navegacional *Llibres* defineix una recuperació d'informació sobre els llibres de la biblioteca, recuperant la

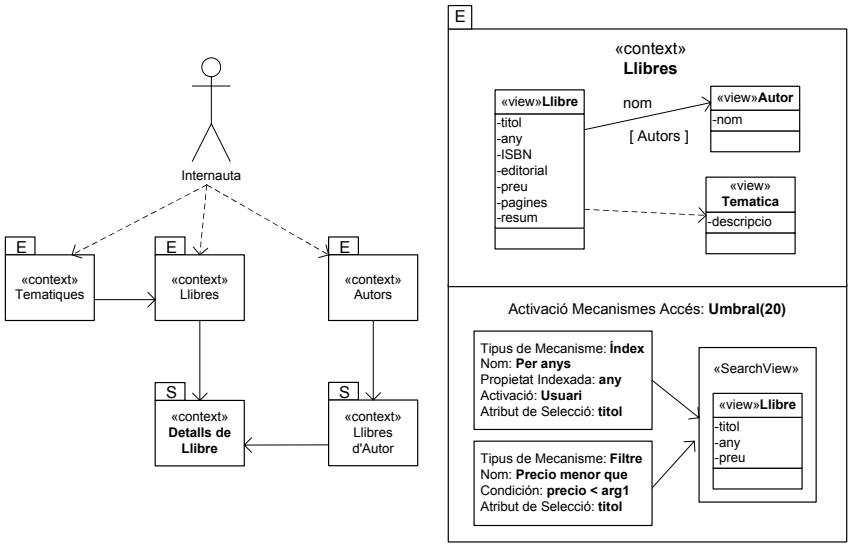


Figura 7.2 – Mapa Navegacional i Contexte *Llibres* de la Biblioteca

seua fitxa (títol, any, ISBN, editorial, preu, pàgines i resum), així com dues relacions navegacionals: una relació de contexte que permet obtenir el nom de l'autor del llibre i una relació de dependència de contexte que recupera la descripció de les temàtiques del llibre.

A més a més, aquest contexte té definits dos mecanismes d'accés: un índex i un filtre. L'índex està definit sobre la propietat any d'un llibre, i el filtre permet buscar llibres atenent al seu preu. Ambdós estructures empren la mateixa vista de resultats que mostra informació resumida sobre els llibres (títol, any i preu).

7.2.1 Patró de Disposició d'Informació

El Patró de Disposició d'Informació permet definir com es disposarà la informació de la vista que defineix un contexte navegacional. Per a

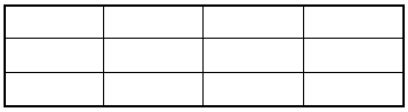


Figura 7.3 – Patró de Disposició Tabular

aconseguir-ho, caldrà etiquetar els elements que defineixen un contexte per descriure aquesta disposició.

Aquest patró pot aplicar-se a:

- la **classe directora**: indica com es presentaran les instàncies d'aquesta classe
- les **relacions navegacionals**: indica com es presentaran les instàncies de les classes complementàries amb respecte a la seua classe orige de la relació. Sols pot aplicar-se sobre relacions navegacionals on la cardinalitat destí siga 1.
- les **vistes de resultats** dels mecanismes d'acces a la informació. S'aplica a la classe directora i les relacions navegacioanls d'aquesta vista.

El Patró de Disposició d'Informació és en realitat un conjunt de patrons (Tabular, Registre, Arbre, Text i Mestre-Detall) que es descriuen a les següents seccions.

7.2.1.1 El Patró de Disposició Tabular

El Patró de Disposició **Tabular** crea una aparença en forma de taula. Representa un requeriment de presentar la informació a visualitzar de manera compacta.

Aquest patró es sol emprar per presentar gran quantitat d'informació, per la seua capacitat de compactar les dades.

Título	Año	Precio	ISBN
El señor de los anillos	1954	24€	540-3456-332X
La isla del tesoro	1970	13€	345-3452-569H
Hem de parlar	2003	18€	84-664-0080-X
...	...	...	...

Disposición Tabular (V)

Título	El señor de los anillos	La isla del tesoro	Hem de parlar	...
Año	1954	1970	2003	...
Precio	24€	13€	18€	...
ISBN	540-3456-332X	345-3452-569H	84-664-0080-X	...

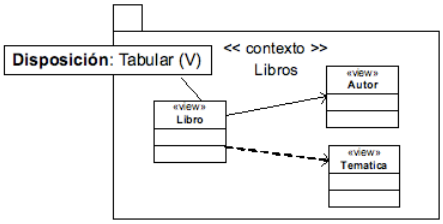
Disposición Tabular (H)

Figura 7.4 – Patró de Disposició Tabular: alternatives

Els atributs i operacions de la classe navegacional afectada pel patró s'organitzen dins la taula. En funció d'aquesta organització, podem tenir dos tipus de disposició tabular:

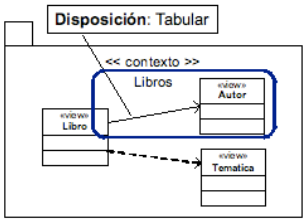
- *Tabular Vertical ó Tabular (V).* La taula creix en files. Cada instància és una nova fila de la taula.
- *Tabular Horitzontal ó Tabular (H).* La taula creix en columnes. Cada instància és una nova columna de la taula.

La Figura 7.5 a la pàgina següent mostra l'aplicació d'aquesta disposició als elements de la classe directora i com aquesta informació hauria de visualitzar-se.



Título	Año	Precio	ISBN
El señor de los anillos	1954	24€	540-3456-332X
La isla del tesoro	1970	13€	345-3452-569H
Hem de parlar	2003	18€	84-664-0080-X
...	...	...	...

Figura 7.5 – Patró de Disposició Tabular aplicat a Classe Directora



Título	Año	Precio	ISBN	Autor Nombre
El señor de los anillos	1954	24€	540-3456-332X	J.R. Tolkien
La isla del tesoro	1970	13€	345-3452-569H	R.L. Stevenson
Hem de parlar	2003	18€	84-664-0080-X	Andreu Buenafuente
...	...	...	...	

Figura 7.6 – Patró de Disposició Tabular aplicat a Relació Navegacional

La Figura 7.6 mostra l'aplicació d'aquesta disposicio a la relació entre les classes navegacionals *Libro* i *Autor*. Aquesta disposició sols pot aplicar-se a aquesta relació i no a la de temàtiques, ja que un llibre pot tenir moltes temàtiques.

La disposició dels atributs de l'autor del llibre amb respecte del llibre es realitzarà en format tabular (com si es tractara d'una propietat més de la classe llibre).

De la mateixa manera, la Figura 7.7 a la pàgina següent mostra l'aplicació sobre una vista de resultat (sobre la classe directora de la vista de resultats).

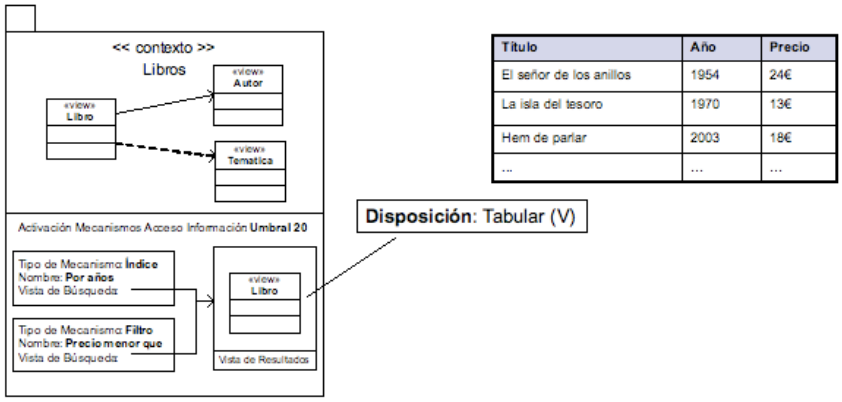


Figura 7.7 – Patró de Disposició Tabular aplicat a la Vista de Resultats

7.2.1.2 El Patró de Disposició en Registre

El Patró de Disposició **Registre** presenta les dades en forma de "registre". Un registre (veure Figura 7.8 a la pàgina següent) és una forma especial de tabular en la que cada instància es mostra de manera independent en un taula horitzontal. Aquest patró sol emprar-se per a presentar informació detallada de poques o una instància, ja que no compacta la informació i requereix de molt espai per presentar moltes instàncies.

La Figura 7.9 a la pàgina següent presenta l'aplicació d'aquest patró sobre la classe directora i com seria la visualització d'instàncies.

Aplicat a la relació entre *Libro-Autor* (relació un-a-un), pot veure's a la Figura 7.10 a la pàgina 177.

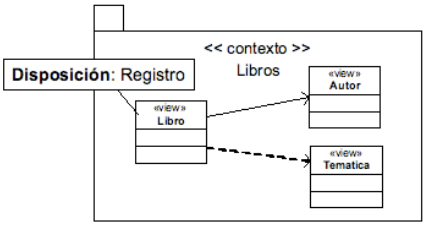
Igualment, pot aplicar-se a la vista de resultats, encara que com a norma general no serà recomanable ja que una vista de resultats haurà de presentar vàries instàncies. La mostra l'efecte d'aplicar-ho a una vista de resultats d'un mecanisme d'accés a la informació.

Título	El señor de los anillos
Año	1954
Precio	24€
ISBN	540-3456-332X

Título	Hem de parlar
Año	2003
Precio	18€
ISBN	84-664-0080-X

Título	...
Año	...

Figura 7.8 – Patró de Disposició Registre



Título	El señor de los anillos
Año	1954
Precio	24€
ISBN	540-3456-332X

Título	Hem de parlar
Año	2003
Precio	18€
ISBN	84-664-0080-X

Figura 7.9 – Patró de Disposició Registre aplicat a la Classe Directora

7.2.1.3 El Patró de Disposició en Arbre

El Patró de Disposició **Arbre** defineix una presentació d'informació en una estructura jerárquica de visualització. Normalment aquest tipus de visualització és molt adient per a relacions d'associació reflexives al diagrama de classes. La Figura 7.11 mostra un exemple d'aplicació d'aquest patró.

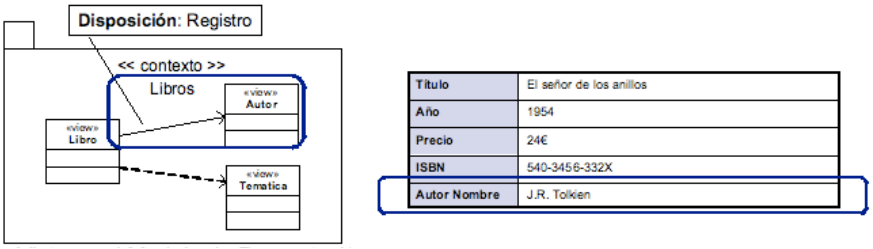


Figura 7.10 – Patró de Disposició Registre aplicat a la Relació Navegacional

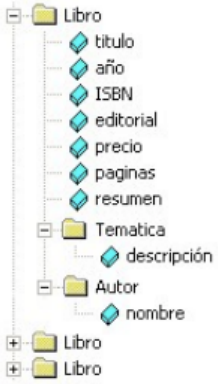


Figura 7.11 – Patró de Disposició en Arbre

7.2.1.4 El Patró de Disposició de Text

El Patró de Disposició **Text** defineix una recuperació d'informació "sense format". Aquest patró és molt útil quan la informació a mostrar està molt clara en el contexte d'ús i no cal enriquir la presentació amb informació sobre els noms dels atributs a mostrar. La Figura 7.12 a la pàgina següent mostra un exemple d'aplicació d'aquest patró de presentació a la classe directora del contexte.

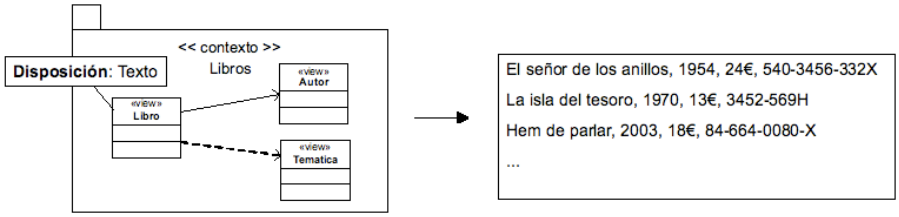


Figura 7.12 – Patró de Disposició Text

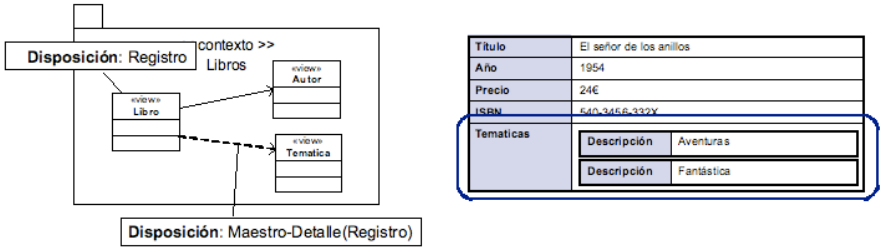


Figura 7.13 – Patró de Disposició Mestre-Detall

7.2.1.5 El Patró de Disposició Mestre-Detall

El Patró de Disposició **Mestre-Detall (P)** s'utilitza per presentar informació aplicada a relacions de navegació amb cardinalitat "molts" amb respecte a la classe origen de la relació.

Aquest patró crea un vincle entre la informació de la classe origen (que juga el paper de mestre) i el de la classe destí (detall) de la relació. Cal especificar-se, obligatòriament el tipus de patró de presentació que s'aplicarà al detall, podent ser qualsevol dels patrons de disposició vist amb anterioritat.

La Figura 7.13 mostra l'aplicació del patró de disposició Mestre-Detall (amb el detall a mode Registre) per al contexte *Libros* i com seria una visualització adient d'aquesta especificació.

7.2.2 Patró d'Ordenació de Dades

El Patró d'Ordenació de Dades permet definir una ordenació sobre la població afectada de manera que la presentació d'aquesta informació seguisca algun criteri d'ordre.

Aquest patró pot aplicar-se a:

- les **classes navegacionals**: indica l'ordre en que es presentaran les instàncies d'aquestes classes
- les **vistes de resultats** dels mecanismes d'accés a la informació: aplicat sobre les classes navegacionals de les vistes de resultats, indica l'ordre en que apareixeran els resultats d'executar un filtre o un índex.

Per tal de definir la ordenació, cal especificar una fòrmula o *expressió d'ordenació*. Aquesta expressió es forma a partir dels atributs de la classe navegacional sobre la que s'aplica, indicant el mode d'ordenació (ascendent o descentent). Les expressions es defineixen com conjuncions de criteris, de la següent manera:

$\texttt{atribut}_1$ `MODE [ AND` $\texttt{atribut}_2$ `MODE [ AND ... ]]`

on `MODE = [ ASC , DESC ]`

La Figura 7.14 a la pàgina següent mostra l'especificació de:

- una ordenació de dades aplicada a la classe navegacional *Libro* del contexte *Libros*: *año DESC AND precio ASC*, indicant que es vol recuperar els llibres ordenats per any descendentment i després per preu ascendentment
- una ordenació de dades aplicada a la classe navegacional *Libro* de la vista de resultats dels mecanismes d'accés, indicant que es vol recuperar els llibres ordenats per preu ascendentment

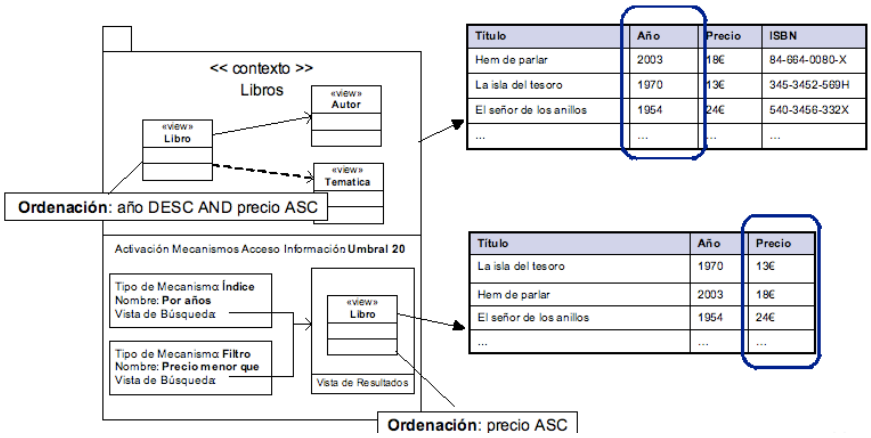


Figura 7.14 – Criteri d'Ordenació

Però de vegades és necessari poder-li proporcionar a l'usuari amb vàries alternatives d'ordenació per tal que ell en puga triar alguna d'aquestes. Es pot donar suport a aquest requeriment extenent el llenguatge d'expressions de la següent manera. Si a les expressions vistes abans les anomenem *expressions d'ordenació bàsiques*, també podrem especificar el següent tipus d'expressions:

$\texttt{exprOrdBàsica}_1$ [`OR` $\texttt{exprOrdBàsica}_2$ [`OR ...`]]

Amb aquesta notació passen a ser alternatives cada expressió d'ordenació addicional, sent la primera l'ordenació per defecte que tindrà l'usuari i que podrà canviar per una altra de les alternatives en qualsevol moment.

Així, per exemple, podriem especificar la següent expressió d'ordenació per a la classe navegacional *Libro*:

año DESC AND precio ASC OR titulo ASC

indicant que l'usuari veurà per defecte els llibres ordenats per any descent i preu ascendent, però podrà canviar i aconseguir veure'ls ordenats alfabèticament.

7.2.3 Patró de Paginació

El Patró de Paginació permet "trocejar" la visualització de dades en *blocs* per tal que siga més accessible per a l'usuari. A diferència d'un mecanisme d'accés a la informació que redueix la quantitat d'informació a mostrar, tractant d'adequar-se a l'interés de l'usuari, aquest patró no fa cap reducció. Simplement organitza tota la població (per gran que siga) per tal que la seua exploració siga més ventajosa. El concepte del **bloc** és el principal d'aquest patró.

Exemples hi ha molts a la xarxa, que responen a aquest patró. Però, el més representatiu avuí en dia per ús pot ser al Google. La Figura 7.15 a la pàgina següent mostra una pàgina resultat d'una búsqueda al Google. Com es pot veure, la quantitat de links a mostrar és molt gran i per això, Google realitza una paginació de dades.

Així doncs, l'objectiu d'aquest patró és poder donar suport a un comportament d'aquest tipus, on hi ha una població a visualitzar i cal que aquesta siga paginada per facilitar-li la tasca a l'usuari.

Aleshores, com s'ha d'aplicar a poblacions, es podrà especificar una paginació sobre:

- la **classe directora**: indica que la paginació es realitzarà sobre els elements d'aquesta classe.
- les **relacions navegacionals**: indica que la paginació es realitzarà sobre els elements de la classe navegacional destí de la relació navegacional
- les **vistes de resultats** dels mecanismes d'accés a la informació. S'aplica a la classe directora i les relacions navegacionals d'aquesta vista.

Per tal d'especificar aquest patró haurem de configurar una sèrie de propietats que es descriuen a continuació.

Figura 7.15 – Google: Exemple de Paginació

La primera propietat, i principal, és el **número d'elements** que formaran cada bloc, que també es sol anomenar *cardinalitat del bloc*. Aquesta cardinalitat pot ser *estàtica* o *dinàmica*. Una **cardinalitat estàtica** indica que el número d'elements dels blocs està fixe i predefinit. Una **cardinalitat dinàmica** indica que el número d'elements dels blocs es variable i pot prendre diferents valors. Associat a aquesta cardinalitat dinàmica es pot indicar una expressió que restringisca els valors possibles en forma d'una enumeració de valors (per exemple: 10, 25, 50,

Figura 7.16 – Patró de Paginació: paginació Seqüencial

100) o d'una condició que ha de complir la cardinalitat (per exemple: cardinalitat ≤ 100).

A més de la cardinalitat, cal especificar el **mode d'accés**, que defineix com es podran explorar els blocs. Hi ha dos tipus d'accés: el *mode seqüencial* i el *mode aleatori*. El **mode seqüencial** especifica que a partir d'un bloc sols podem accedir a l'immediatament anterior i a l'immediatament posterior. El **mode aleatori** indica que podem accedir a qualsevol bloc des de qualsevol bloc.

Finalment podem marcar la paginació com **circular** per convertir la colecció ordenada de blocs en un búfer circular, és a dir, definir el primer bloc com el següent bloc de l'últim bloc i definir l'últim bloc com l'anterior bloc del primer bloc.

La Figura 7.16 mostra un exemple de paginació aplicada a la classe navegacional *Libro*. Aquesta paginació està definida amb una cardinalitat de 10 elements, estàtica i d'accés seqüencial. És per això que la població de llibres es veurà trocejada de 10-en-10 llibres, l'usuari no podrà canviar aquesta cardinalitat de 10 i des de cada pàgina sols tindrem accés a la següent i a l'anterior.

Figura 7.17 – Patró de Paginació: paginació Aleatòria

Un altre exemple, però aquesta vegada amb cardinalitat de 4 elements i accés aleatori. La Figura 7.17 mostra un exemple d'implementació d'aquest requeriment.

7.2.4 Patró d'Ordre d'Aparició

El Patró d'Ordre d'Aparició indica una relació en l'ordre d'aparició dels atributs navegacionals i operacions navegacionals d'un contexte. En altres contextos, a aquesta propietat se la sol anomenar "*tabbing*" .

Aquest patró s'aplica a:

- els **atributs navegacionals**
- les **operacions navegacionals**

Associat a cadascun d'aquestos elements podem etiquetar amb un **ordre d'aparició** (representat mitjançant nombres enters), indicant una relació d'ordre entre aquestos elements en un contexte navegacional o una vista de resultats.

La Figura 7.18 a la pàgina següent mostra l'especificació d'aquest ordre sobre el contexte navegacional *Libros* i com afecta aquesta propietat en la implementació. Com es pot veure, el primer que apareix és el títol del llibre, després el seu autor (el nom), les pàgines, l'ISBN, etc.

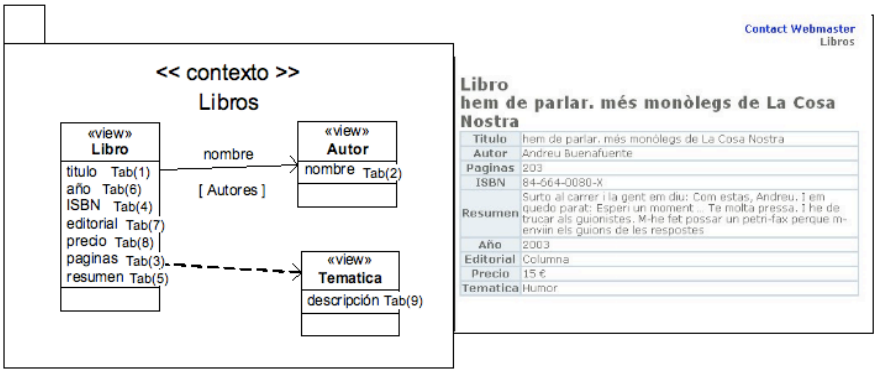


Figura 7.18 – Patró d'Ordre d'Aparició

7.3 Conclusions

En aquest capítol s'ha presentat un mecanisme per introduir una sèrie de patrons de presentació a nivell de modelatge. La idea principal d'aquestos patrons és la de capturar certs requeriments abstractes per a presentar les unitats de visualització definides al Model Navegacional.

Amb els *Patrons de Disposició* es defineix com s'ha de presentar una informació amb respecte d'una altra, mentre que el *Patró de Paginació* representa un requeriment d'accessibilitat a la informació dividint la visualització en conjunts de població.

El *Patró d'Ordenació de Dades* permet definir criteris d'ordenació que s'aplica a la població dels nodes a l'hora de visualitzar la informació.

Finalment, el *Patró d'Ordre d'Aparició* captura la importància de la informació definint quina informació ha d'aparèixer abans i quina no.

No és objectiu d'aquest descriure la manera concreta en que s'ha de visualitzar la informació, ni tan sols el que aquesta interfície es descompondrà en components que s'hauran de combinar, ni tampoc el fet d'introduir aspectes de disseny a la interfície. Com aquestos

aspectes de disseny són introduïts al mètode pot veure's al Capítol 8. L'objectiu doncs d'aquest model, com s'ha esmentat abans, és establir restriccions rellevants a tenir en compte per desenvolupar l'interfície Web de l'aplicació final.

Capítol 8

Implementació de la Interfície Web

8.1 Introducció

En l'actualitat, existeixen moltes estratègies per abordar la implementació d'una aplicació web. Hi ha nombrosos llenguatges de programació, frameworks d'implementació, eines de disseny, etc..

Malgrat açò, l'objectiu que es persegueix en aquest treball és fer-ho de manera deslligada a cap tecnologia, però que siga aplicable a qualsevol. En realitat, la tasca no és del tot complicada ja que arribem a aquesta fase tenint com entrada models conceptuals, independents de plataforma tecnològica. Per a aconseguir-ho, primer s'ha realitzat un anàlisi a nivell conceptual de com haurien de ser les pàgines web, i després s'ha definit una estratègia d'implementació d'aquestes idees conceptuals.

Per abordar l'anàlisi inicial de les pàgines web, observem les implementacions existents en diferents aplicacions web per intentar abstaure els diferents continguts i tipus de pàgines que conformen l'aplicació. Així, com veurem més endavant, podrem observar que existeixen pàgines en-

carregades de la gestió d'informació i que permeten accedir a l'execució de serveis, altres que permeten estructurar la navegabilitat dins l'aplicació, etc.

Una vegada detectats els tipus de pàgines que poden constituir una aplicació web i com s'organitzen, tractarem d'analitzar els tipus de continguts que solen apareixer en aquestes pàgines. Així, per exemple, podrem adonar-nos que existeixen zones de la pàgina web encarregades de proporcionar un conjunt d'enllaços a altres pàgines, zones que són responsables d'informar a l'usuari de la seqüència de pàgines que ha anat navegant fins arribar a l'actual, altres que mostren informació sobre l'empresa que hi ha per darrere de l'aplicació, etc.

Prenent aquesta informació com punt de partida, es proposarà un mode de conceptualitzar una pàgina web com un conjunt de zones de contingut, en funció del tipus de pàgina. A més a més, emprarem una sèrie de principis de qualitat per tal d'assegurar quins continguts són més adients, necessaris o dolents dins de cada tipus de pàgina. L'objectiu és apropar-nos a aconseguir aplicacions final de major qualitat.

Finalment, es defineix una estratègia per a la implementació d'aquestos diferents tipus de pàgines emprant com "seccions" aquestes zones de contingut. D'aquesta manera, la solució proposta per desenvolupar la capa d'interfície de les aplicacions web empra constructors de major granularitat que les meres etiquetes o marques HTML sobre les que es sol mapejar.

D'altra banda, avuí en dia, el Desenvolupament Dirigit per Models i MDA [86] estan guanyant popularitat com una manera possible i adequada per desenvolupar programari en diferents entorns, inclós el Web. Com a conseqüència, diferents eines tant des d'un punt de vista acadèmic com industrial, estan oferint els seus propis processos MDA per produir aplicacions Web.

En aquesta tesi s'introdueix un entorn a l'estil MDA per al desen-

volupament d'aplicacions web, que segueix l'aproximació proposada pel mètode OOWS. OOWS constitueix una extensió del mètode OO-Method [93] per donar suport al modelatge i desenvolupament d'aplicacions web. Des d'un punt de vista MDA, tant OO-Method com OOWS defineixen el seu PIM a partir del llenguatge d'especificació que defineixen les seues primitives de modelatge conceptual. La solució proposta en aquest treball integra l'eina industrial *OlivaNOVA* [23], com a implementació en àmbits industrials del mètode OO-Method, i desenvolupada per l'empresa CARE Technologies S.A. [26]. Aquest entorn *OlivaNOVA* permet generar de manera automàtica el codi de l'aplicació, emprant transformacions model-a-codi, segons MDA. OOWS extén aquest procés de generació per obtenir l'interfície web que s'integre amb les aplicacions generades per *OlivaNOVA*.

En aquest capítol es presenta una estratègia de desenvolupament que dóna una solució concreta a aquestes necessitats en dos passos: primer es defineix una proposta de desenvolupament de les pàgines web, i segon és realitza un mapeig entre els conceptes (primitives) del mètode i aquestes pàgines web.

En resum, els objectius desitjats per abordar el desenvolupament de les aplicacions web són els següents:

- Proposar un marc conceptual per al desenvolupament de pàgines web
- Estudiar una classificació de les pàgines web en funció del seu objectiu
- Analitzar els diferents continguts que poden aparèixer en les pàgines web i a què es deuen
- Avaluar una estratègia d'implementació d'aplicacions web basada en pàgines web predisenyades

- Aplicar des d'un principi bons criteris de qualitat en el desenvolupament de les pàgines web que conformen l'aplicació web
- Definir un conjunt de transformacions entre els conceptes del domini *(espai del problema)* i els conceptes en una solució tecnològica concreta *(espai de la solució)*.

8.2 Estructura Conceptual de les Pàgines Web

Per començar devem analitzar aplicacions web centrant-nos en estudiar els tipus de pàgines que apareixen i quins continguts aporten. Si determinem el conjunt de pàgines formades per un conjunt de continguts, contruir una aplicació web no serà més que emprar aquestes seccions combinades per desenvolupar l'aplicació web final.

Aquesta manera d'entendre allò que és una pàgina web permet veure quin tipus de responsabilitat ha de tenir cada pàgina, abans d'abordar la implementació de l'aplicació, decidint, per exemple, que una pàgina d'inici deu ser una pàgina que estructure l'accés dins l'aplicació, ja que es tracta de la primer pàgina que es visita i per tant el punt d'entrada a qualsevol part del sistema.

També es poden prendre decisions de l'estil de: "després d'una pàgina d'estructura de navegació deu aparèixer una pàgina que proporcione contingut d'informacio". D'aquesta manera, podem dissenyar prèvia la implementació, quina serà la navegabilitat pel sistema, podent estudiar i millorar, si fora necesari, l'accessibilitat dins l'aplicació.

Suposem per exemple que estem en una tenda per a la venta de llibres. Per motius de fomentar l'ús de l'aplicació i potenciar les ventes volem que els usuaris que es conecten puguen trobar el llibre concret que busca en menys de 3 clics de navegació. Fent un disseny de l'estructura navegacional, podrem determinar que l'aplicació l'anem a desenvolupar seguint aquesta restricció o deurem fer algun retoc en el disseny.

Altre avantatge d'aquesta aproximació és la possibilitat d'obtenir una ràpida prototipació. Si som capaços de parametritzar un conjunt de tipus de pàgines seguint unes pautes preestablertes, construir un prototipus d'una aplicació es constituirà d'enllaçar adequadament aquestes pàgines i triar els continguts que deuen apareixer en cada tipus de pàgina en aquesta aplicació.

8.2.1 Taxonomia

En aquesta secció s'analitzen els diferents tipus de pàginas que existeixen en la majoria de les aplicacions web actuals. El nostre criteri de classificació de pàgines l'anem a centrar en la responsabilitat que té cada pàgina amb respecte a l'aplicació en sí.

Per aconseguir, anem a repasar quin és el model de funcionament d'una aplicació per a la web típica:

- un usuari entra en un lloc web (de manera anònima o identificant-se) i obté la pàgina d'inici. Aquesta pàgina proporciona accés a la resta de pàgines que conformen l'aplicació. En la majoria dels casos, podem considerar aquest tipus de pàgines inicials com pàgines destinades a estructurar l'accés. També és habitual que en aquestes pàgines d'inici aparega informació institucional de l'empresa (nom, adreça, informació de contacte, logotipus, etc.).

- si seleccionem un enllaç, arribem a altres pàgines en les que es mostra informació sobre els objectes d'interés (productes, personal, etc.). Són pàgines encarregades de proporcionar contingut d'informació. En aquestes pàgines també pot apareixer l'enllaç a l'execució d'alguna funcionalitat, com per exemple, sol·licitar una comanda de material, informar de l'interés sobre un determinat producte, reservar alguns recursos, etc. Dins d'aquestes pàgines, sol ser habitual el que es puga navegar a altres pàgines seleccionant

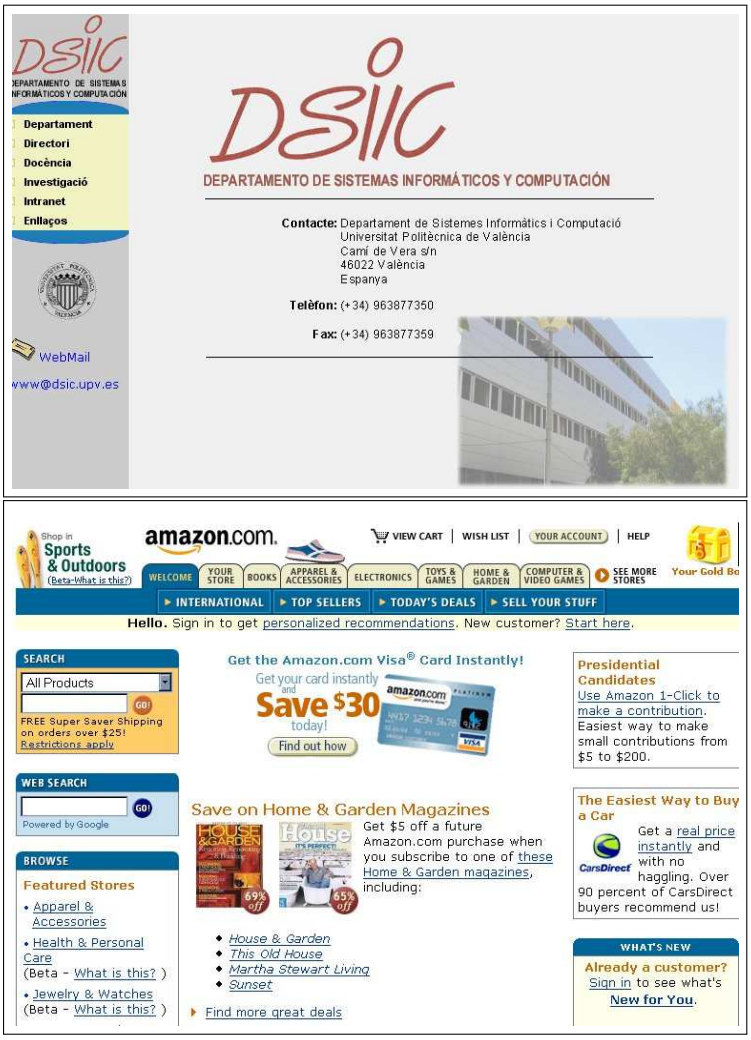


Figura 8.1 – Exemple de Pàgines "*Home*"

algun enllaç sobre certa informació que apareix en la pàgina. Quan seleccionem algun d'aquestos enllaços, l'aplicació canvia la pàgina per una altra on es mostra el contingut d'informació relacionat amb el que hem seleccionat.

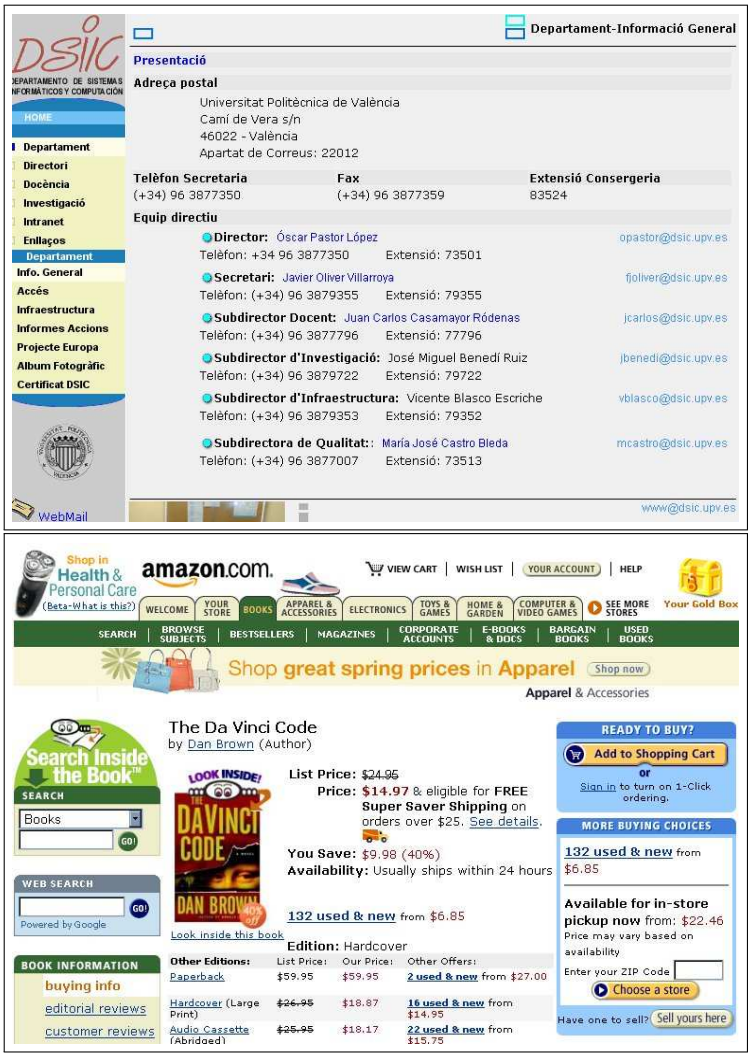


Figura 8.2 – Exemples de Pàgines Web amb Contingut d'Informació

- com hem dit abans, a partir d'una pàgina de contingut d'informació també podem llançar l'execució d'alguna funcionalitat. Quan açò esdevé, el sistema proporciona a l'usuari pàgines on apareix un o

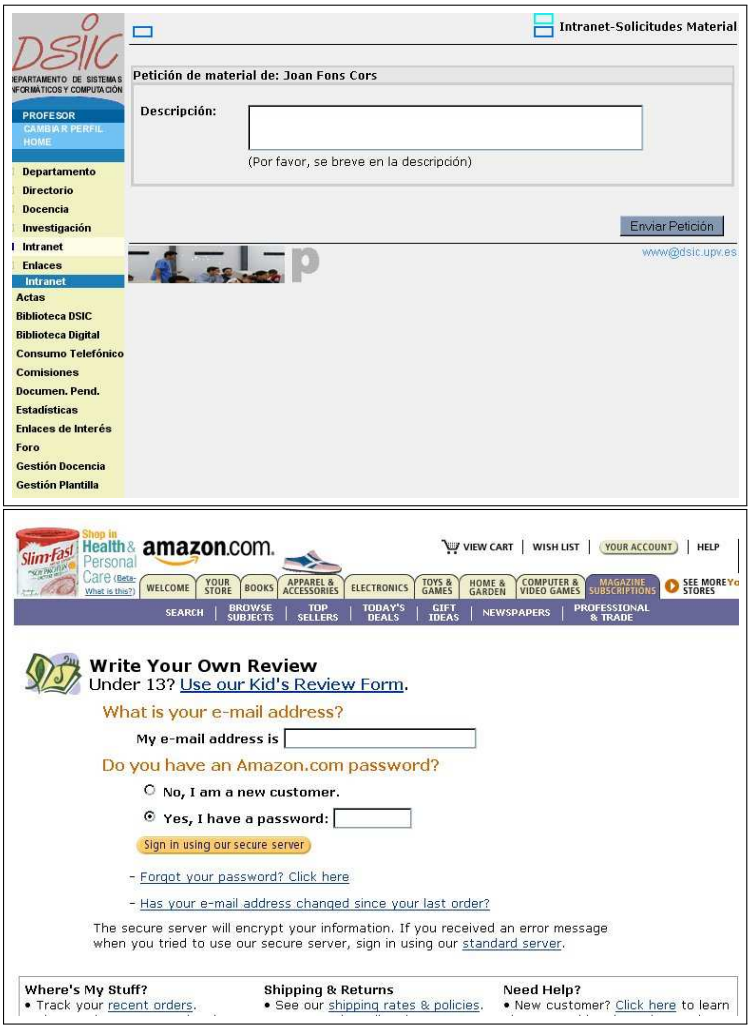


Figura 8.3 – Exemples de Pàgines Web d'Execució de Funcionalitat

diversos formularis per a la introducció de dades (arguments) per a la execució d'aquesta funcionalitat, afegint els botons de "Acceptar" (executar) i "Cancel·lar" (abortar l'execució del servei).

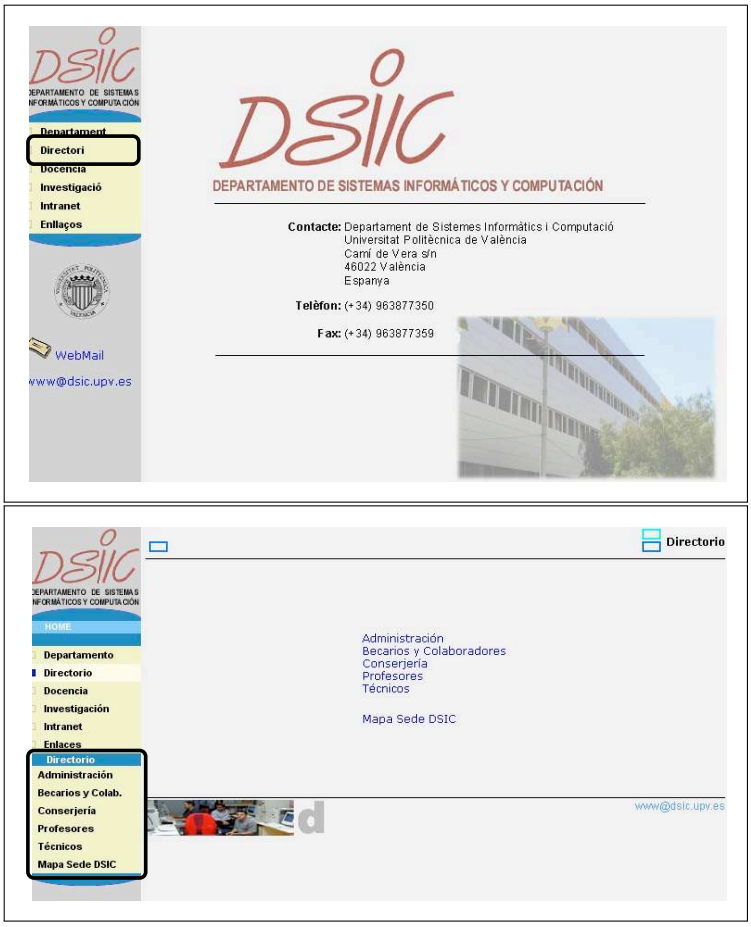


Figura 8.4 – Exemple de Subsistema de Navegació

- una altra possibilitat és navegar a pàgines web que tenen com objectiu proporcionar accés a una part de l'aplicació que fora d'aquesta pàgina no es tenia. Són com les pàgines d'inici, pàgines que estructuren l'espai navegacional introduint nous enllaços a altres pàgines web, facilitant la navegació dins el sistema.

- també hi ha pàgines resultat de la invocació d'un criteri de selecció

Figura 8.5 – Exemple Pàgina Web amb Estructura d'Indexació

o filtre. Habitualment aquestes pàgines proporcionen un conjunt limitat d'entrades que solen donar accés a les pàgines de contingut d'informació per mostrar informació més detallada sobre l'ítem seleccionat.

Encara que pogueren existir més tipus de pàgines dins una aplicació web, anem a centrar-nos en l'estudi d'aquestes i detallar les seues responsabilitats. Si volguèrem introduir nous tipus de pàgines dins la proposta, deuriem seguir el mateix raonament que amb aquestos tipus de pàgines.

8.2.1.1 Pàgines d'Informació

Aquest tipus de pàgines són les encarregades de proporcionar a l'usuari certa informació i funcionalitat. L'objectiu d'aquestes pàgines és informar a l'usuari de certa part de l'estat del sistema, permitint-li modificar aquest estat executant alguns serveis que li estan disponibles i permitir-li navegar a altres pàgines seleccionant alguna informació que en ella

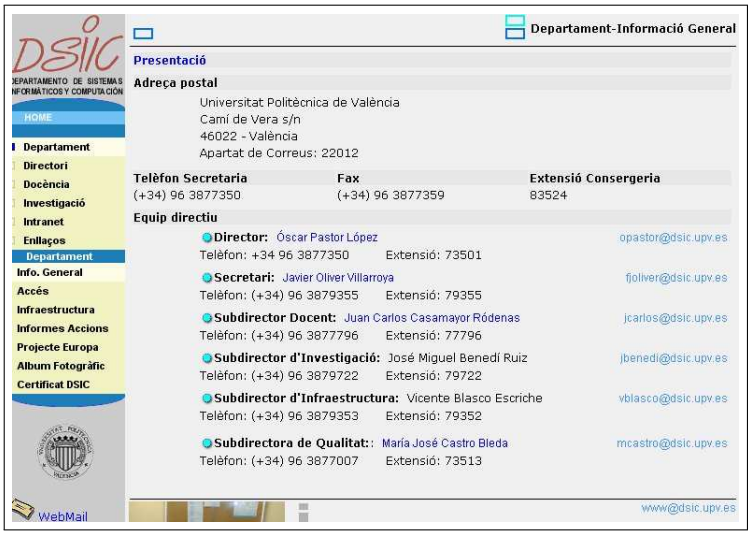


Figura 8.6 – Pàgina Web de tipus Informació

aparega.

Són pàgines de contingut d'informació les següents pàgines (dins el funcionament normal d'una aplicació web):

- les pàgines que exploren la Base de Dades proporcionant informació sobre les entitats que allí hi apareixen
- les pàgines que són conseqüència de l'execució de filtres de selecció d'informació, que presenten el conjunt resultant d'instàncies i enllacen a altres pàgines (també d'aquest tipus) que mostren informació detallada sobre l'ítem seleccionat

8.2.1.2 Pàgines d'Estructuració de la Navegació

Aquest tipus de pàgines són pàgines especialitzades en estructurar la navegabilitat dins l'aplicació, donant accés a pàgines que sols són accessibles a partir d'aquestes pàgines.

Figura 8.7 – Pàgina Web de tipus Estructura de Navegació

Són pàgines d'aquest tipus:

- les pàgines d'inici, que proporcionen accés inicial i estructurat a les diferents parts/seccions de l'aplicació.
- pàgines que seccionen i agrupen altres pàgines per facilitar la navegabilitat dins el sistema.

8.2.1.3 Pàgines d'Entrada de Dades

Aquest tipus de pàgines sorgeixen quan l'usuari sol·licita o invoca l'execució de cert servei i existeix la necessitat d'introduir paràmetres per dur-lo a terme. Habitualment aquestes pàgines contenen formularis d'entrada de dades adequats a les necessitats del servei.

Són pàgines d'aquest tipus:

- les pàgines ques són conseqüència de l'execució de cada servei que deu proporcionar el sistema
- les pàgines de login d'usuari, on es demana informació d'indentificació de l'usuari, identificador i contrassenya

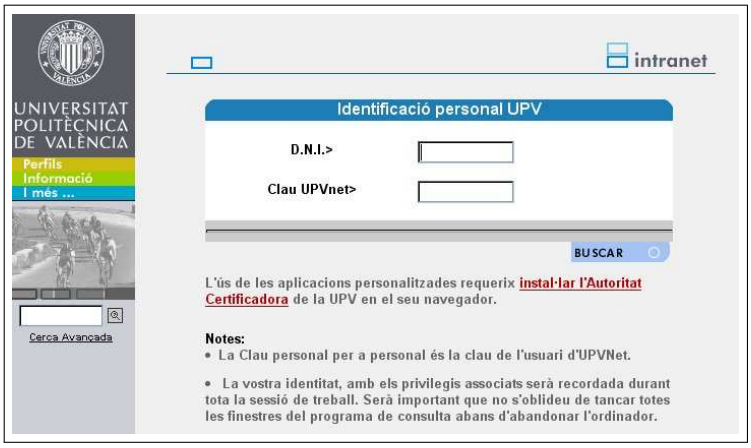


Figura 8.8 – Pàgina Web per a l'Entrada de Dades

- les pàgines de registre d'usuaris
- les pàgines que demanen la condició de filtrar sobre els continguts d'informació

8.2.2 Tipus de Continguts

Podriem pensar que cada tipus de pàgina té uns continguts totalment diferents amb els continguts d'altres pàgines. Tanmateix, si analitzem els desenvolupaments actuals, podrem observar que en la majoria dels casos, existeixen troços de pàgines que es repeteixen en algunes pàgines, i altres que inclús es repeteixen en totes.

Així, per exemple, la majoria de les pàgines web proporcionen una zona en la que es mostren els enllaços de navegació a altres pàgines que es poden alcançar des de l'actual. Una altra zona molt habitual és la que mostra informació sobre l'empresa, i que sol aparèixer en totes de les pàgines.

En funció d'aquest tipus de contingut, podem inclús trobar pàgines que tenen més d'una zona amb contingut del mateix tipus. Així, per

exemple, podem trobar pàgines web que tenen més d'una zona de navegació, estructurant i donant diferents possibles opcions de navegació organitzades per algun criteri. La Secció 8.2.3 analitarà aquesta possible distribució de zones de contingut en funció del tipus de pàgina.

Si seguim analitzant amb més detall, podrem observar inclús patrons d'estructuració i presentació de continguts que solen ser habituals. Els dos exemples més representatius són els següents:

1. Una pàgina web dividida en dues seccions: una més menuda, a l'esquerre, on apareixen un conjunt d'opcions de navegació i una més gran, a la dreta, en la que s'organitza la següent informació, de dalt cap abaix: nom o identificació de la pàgina, ubicació dins el camí navegacional (no sempre), contingut d'informació (zona central dreta, normalment de major proporció), informació sobre la institució (aquesta informació pot apareixer disseminada per la pàgina) i opcionalment més enllaços de navegació (presentats d'una manera compacta)

2. Una pàgina web sense divisions, en la que, de dalt a baix apareix: un conjunt d'enllaços (normalment en disposició horitzontal), identificació de la pàgina, ubicació en el camí navegacional, contingut d'informació i informació institucional.

La proposta tracta de descriure un conjunt de zones de contingut (cadascuna amb un objectiu clar) que després podrem utilitzar per definir els diferents tipus de pàgines. Les següents subseccions tractaran de presentar els tipus de continguts més habituals en els desenvolupaments actuals.

8.2.2.1 Zona d'Informació

Zona encarregada d'accedir a la base de dades, i disposar de manera convenient la informació i funcionalitat que proporciona aquesta pàgina.

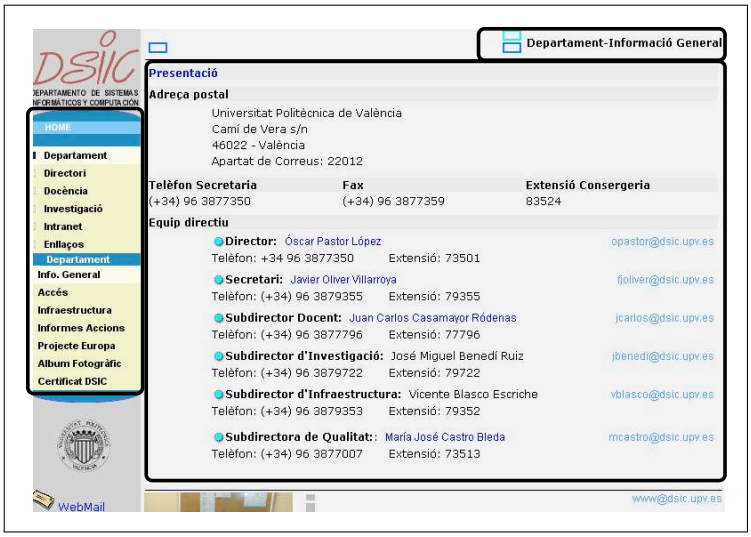


Figura 8.9 – Pàgina Web DSIC: Delimitació de Zones Principals

8.2.2.2 Zona de Navegació

Zona encarregada de proporcionar a l'usuari un conjunt d'enllaços de navegació.

8.2.2.3 Zona d'Ubicació

Zona encarregada de notificar a l'usuari de la ubicació d'aquesta pàgina dins l'aplicació, indicant el camí navegacional seguit o seqüència de pàgines web que s'ha seguit fins arribar a l'actual.

8.2.2.4 Zona d'Informació d'Usuari

Zona responsable de donar-li informació personal (identificar-lo com usuari actiu en el sistema). Aquest tipus de zona sols té sentit per als usuaris que necesiten identificar-se per accedir al sistema.

Figura 8.10 – Pàgina Web Amazon.com: Delimitació de Zones Principals

8.2.2.5 Zona Institucional

Zona encarregada de proporcionar a l'usuari informació sobre la institució, empresa, organisme, etc. que està al darrere de l'aplicació. Habitualment és informació estàtica que es refereix al nom de la institució, adreça, logotipus, contacte, etc.

8.2.2.6 Zona d'Entrada de Dades

Zona encarregada de proporcionar formularis per a la introducció de dades, validar-los i enviar-los per al seu processament.

8.2.2.7 Zona d'Enllaços d'Aplicació

Zona on apareixen enllaços a la funcionalitat comú a totes les aplicacions web: login/logout, home, etc.

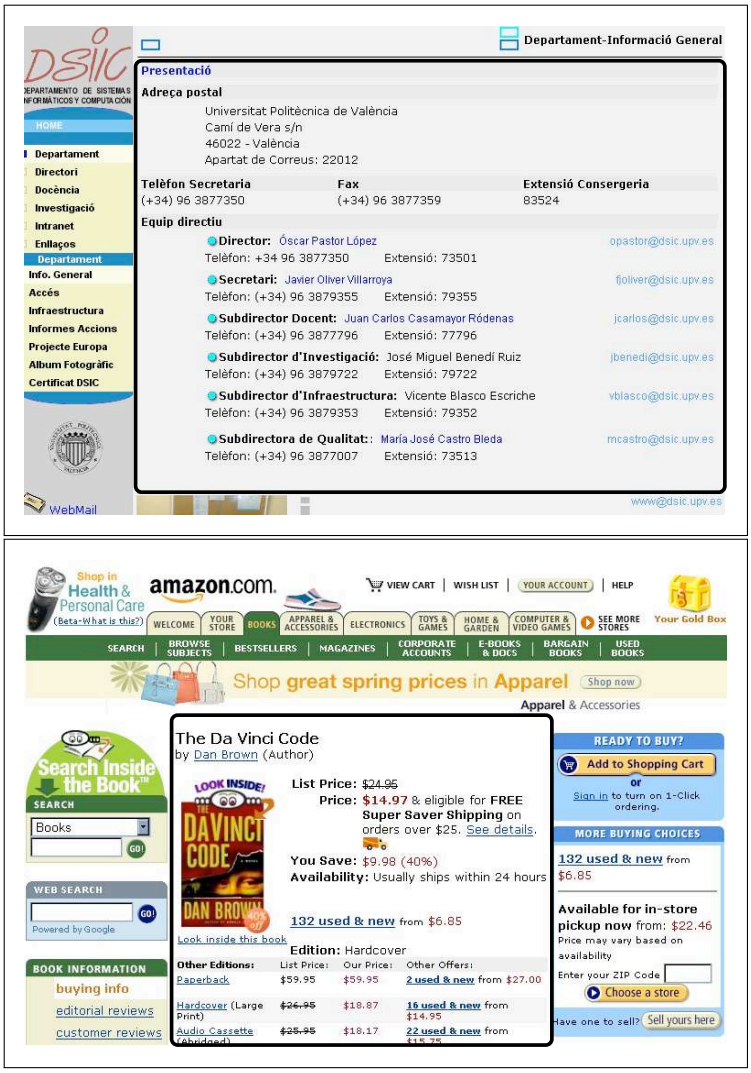


Figura 8.11 – Exemple de Zones d'Informació

8.2.2.8 Zona de Personalització

És un tipus especial de zona d'informació en la que la informació que es recuperarà està personalitzada per a l'usuari actiu que està conectat.

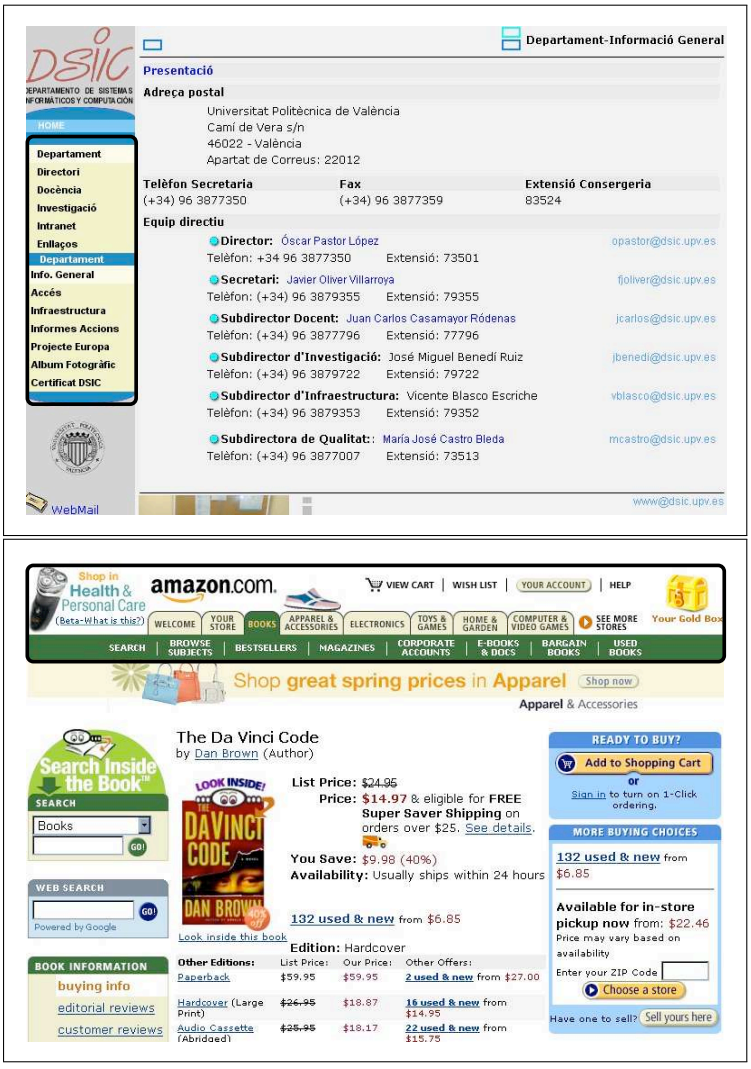


Figura 8.12 – Exemple de Zones de Navegació

Aquesta zona sols té sentit per a usuaris registrats. El seu ús habitual es proporcionar als usuaris ofertes i/o suggerències sobre productes/novetats del sistema que li puguen ser d'interés. Serveix per implementar

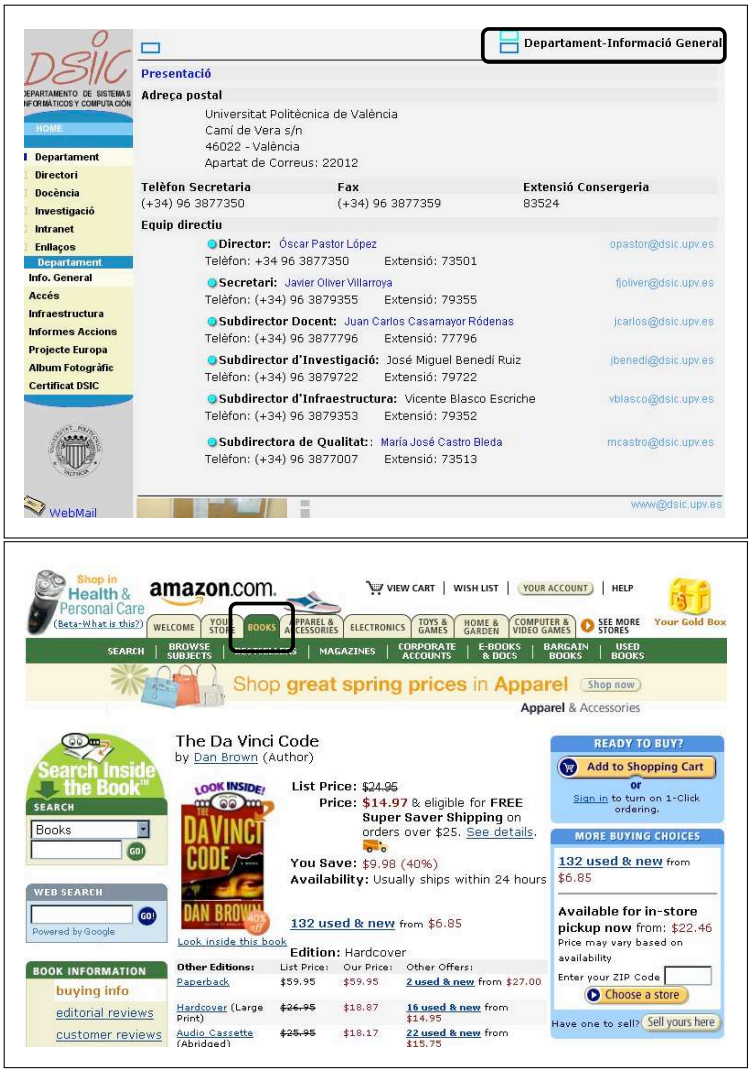


Figura 8.13 – Exemple de Zones d'Ubicació

polítiques de personalització.

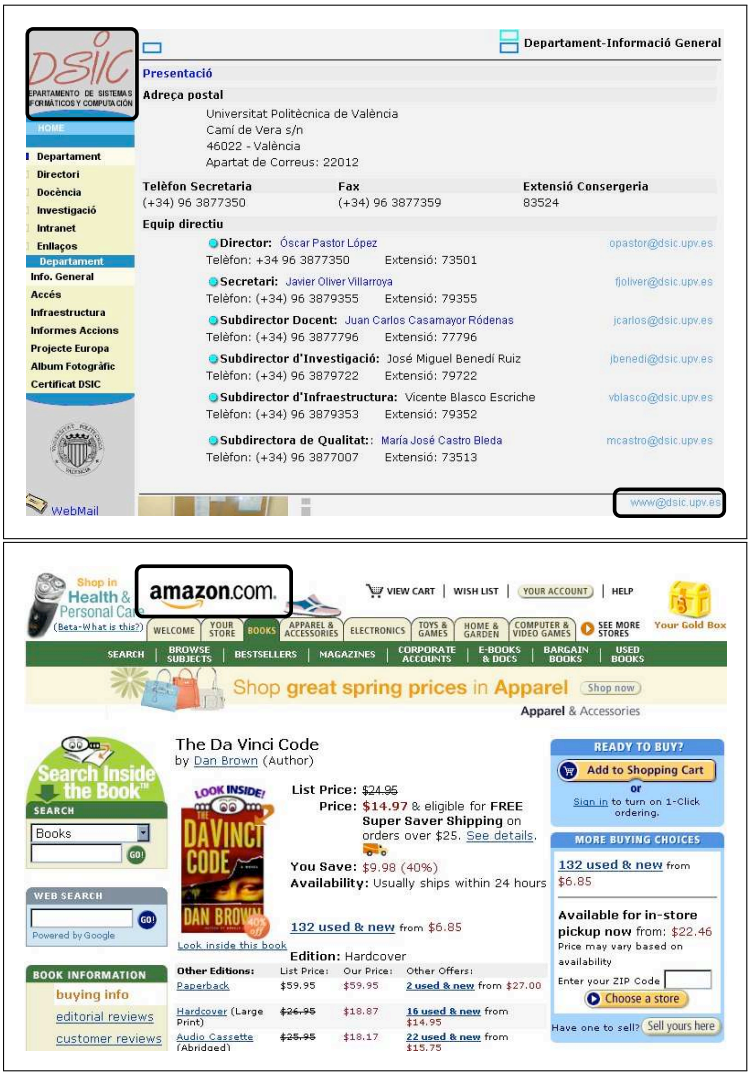


Figura 8.14 – Exemple de Zones Institucional

8.2.2.9 Zona d'Estructures d'Accés

Zona que conté els mecanismes avançats d'exploració dins d'una pàgina. En ella apareixen els filtres de búsqueda, estructures d'indexació, etc.

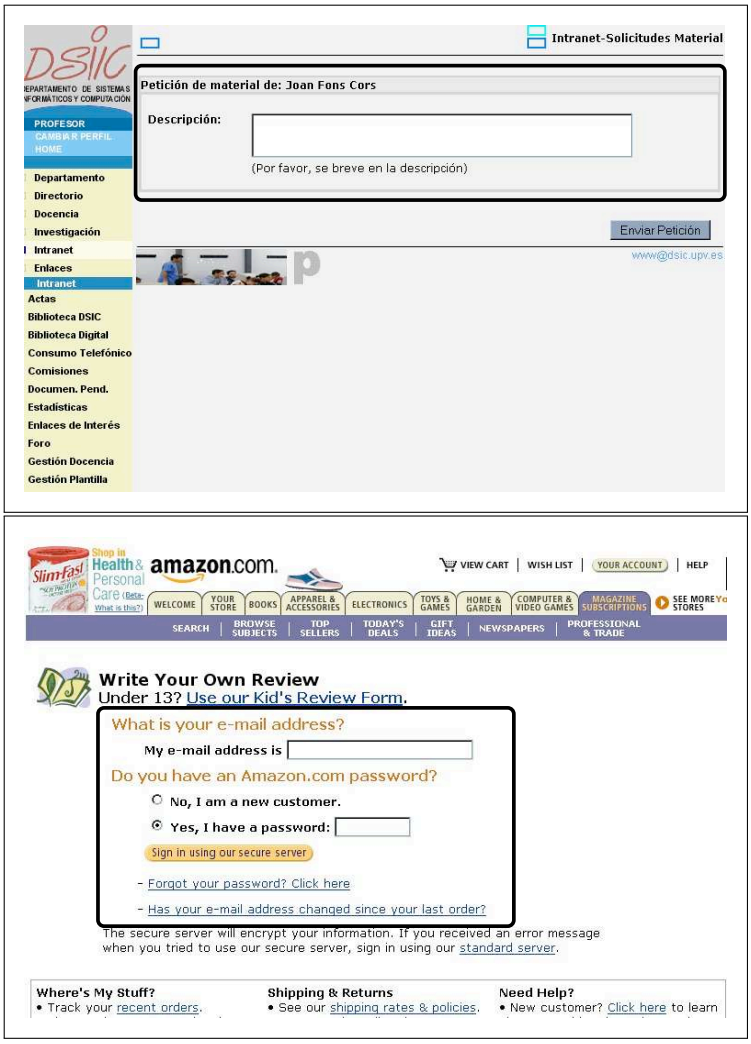


Figura 8.15 – Exemple de Zones d'Entrada de Dades

8.2.2.10 Zona Lliure (Custom)

Zona pensada per afegir, en cas necessari, qualsevol altre tipus de contingut no censat anteriorment, donant una major flexibilitat a l'hora

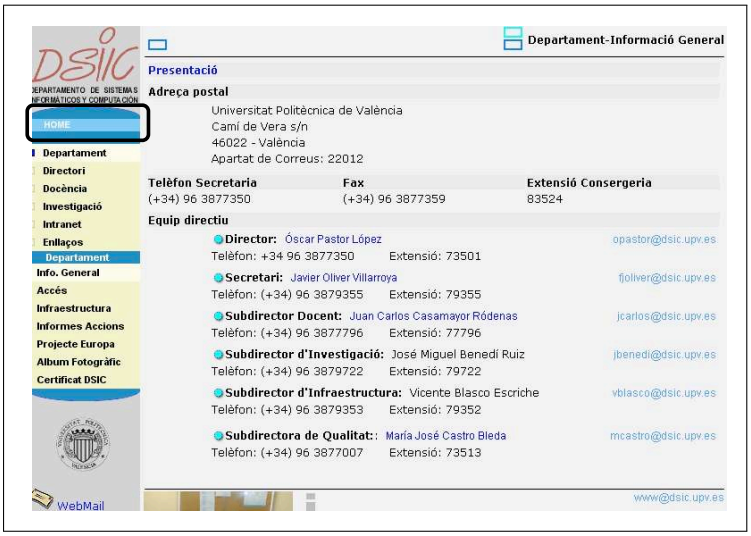


Figura 8.16 – Exemple de Zones d'Enllaços Aplicació

Figura 8.17 – Exemple de Zones de Personalització

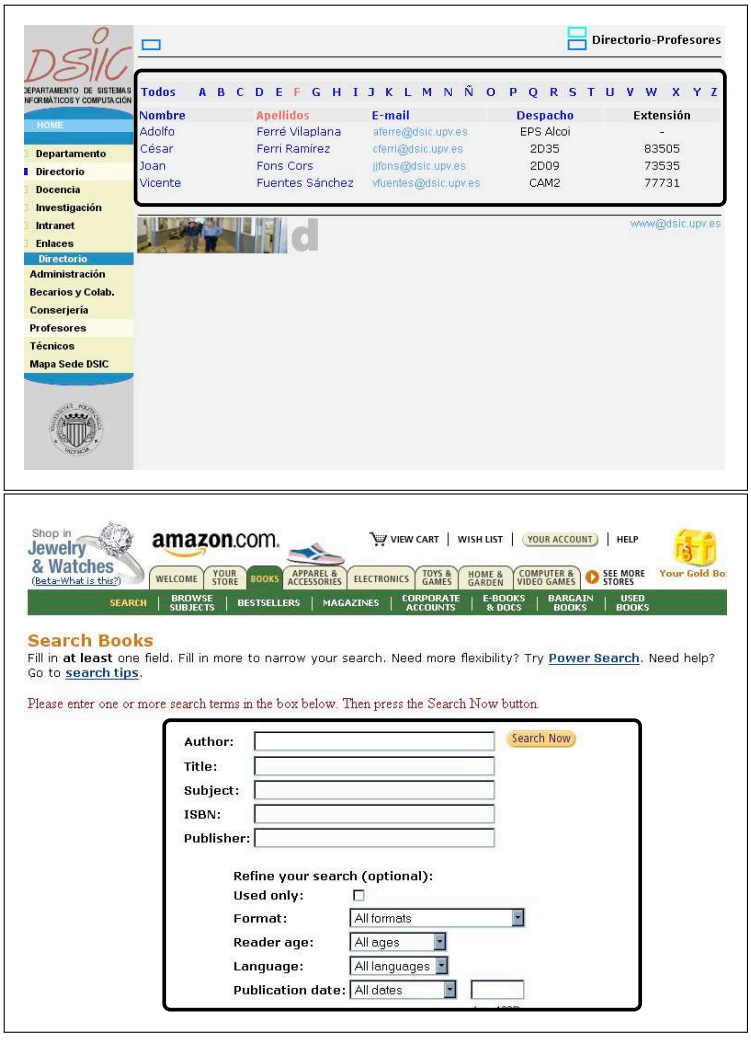


Figura 8.18 – Exemple de Zones d'Estructures Accés

de construir l'aplicació.

8.2.3 Estructura Conceptual

Una vegada detectats els tipus de pàgines i les zones de contingut que poden aparèixer en les diferents pàgines, devem establir una sèrie de regles que ens indiquen quines zones de continguts deuen aparèixer obligatòriament en cada tipus de pàgina, quines zones són recomanables, quines són opcionals i quines no és recomanable que apareguen. Aquestes decisions deuen prendre's tenint en compte, quan siga possible, criteris per millorar la qualitat de les aplicacions desenvolupades [88].

Una de les regles per assegurar una major qualitat de les pàgines web és proporcionar a l'usuari *sempre* informació que li indique *on està*, *com ha arribat* fins ací (camí navegacional) i *on pot anar* a partir d'ací. És a dir, d'alguna manera s'obliga a que en tota pàgina web apareguen les zones de navegació i ubicació.

Altra regla és la de no sobrecarregar en excés les responsabilitats d'una pàgina web. Per raons de comprensibilitat, devem evitar desenvolupaments en el que el producte resultant siga farragós i complicat. En funció del tipus d'aplicació i audiència (usuaris potencials) que puguen emprar l'aplicació, és recomanable la separació de continguts en diferents pàgines per tal de no sobrecarregar-les, simplificant i millorant substancialment la usabilitat final de l'aplicació. Malgrat açò, no sempre serà possible determinar el nivell a partir del qual es considera "sobrecarregada" una pàgina, i més tenint en compte la tendència actual on en molts portals web aparèixen agregadors de continguts de molt diferent índole (com s'analitza a la Secció 6.3, al Capítol 6).

A continuació es presenta una taula on apareixen les zones de contingut per un costat i els tipus de pàgines web per altre, indicant les relacions entre ells:

	Pàg. Informació	**Pàg. Estruc. Navegació**	**Pàg. Entr. Dades**
Z. Navegació	✓	✓, ∗	✓
Z. Ubicació	✓	✓	✓
Z. Informació	✓, ∗	X	X
Z. Usuari	+	+	+
Z. Institucional	?, ∗	?, ∗	?, ∗
Z. Enllaços Aplic.	+	+	+
Z. Entrada Dades	X	X	✓
Z. Personalització	?, ∗	?, ∗	?, ∗
Z. Estruc. Accés	?, ∗	?, ∗	?, ∗
Z. Custom	?, ∗	?, ∗	?, ∗

Taula 8.1 – Estructura Conceptual d'una Pàgina Web

✓- obligatori +- recomanable X- no recomanable
?- opcional ∗- poden aparèixer varies

8.3 Estratègia d'Implementació d'una Pàgina Web

La estratègia per a la implementació de les pàgines web que anem a seguir pren com punt de partida el plantejament ja realitzat de categoritzar el tipus de pàgines i detectar el conjunt de zones que van a constituir la pàgina web.

A més a més, haurem de tenir en compte el comportament desitjat d'una pàgina web dins l'aplicació: és desitjable que quan un usuari navegue dins d'una aplicació, les pàgines que la constitueixen li vagen proporcionant informació global sobre les pàgines que ja ha visitat, els continguts que explora, les possibilitats de navegació, etc.

És per açò que deurem considerar que les pàgines web no estàn aïllades unes de les altres, i que, per tant, s'hauran d'anar passant

informació unes a les altres sobre les accions que l'usuari està realitzant i l'estat navegacional en el que es troba actualment. Alguns exemples d'aquesta informació passada d'unes pàgines a altres poden ser els identificadors dels objectes seleccionats, la seqüència de pàgines que l'usuari ja ha navegat, els valors introduïts en un formulari per executar una operació, etc.

8.3.1 Estructura d'Implementació

Tenint en compte aquestes idees bàsiques, anem a distingir dos zones clares dins una pàgina web:

- El PREÀMBLE: troç inicial de la pàgina web encarregada de recollir la comunicació entre les pàgines (arguments d'entrada de la pàgina), provocar el canvi navegacional atenent a l'estat actual, controlar l'accés dels usuaris (sols aquell que tinga permís, etc.
- El CÓS: defineix el contingut de la pàgina. Està compost per les diferents zones de contigunt que formen la pàgina.

El següent troç de codi presenta l'esquelet d'una pàgina web que s'empra per implementar aquestos principis:

```
<!-- PREAMBLE DE LA PÀGINA -->
...
<!-- FI PREAMBLE DE LA PÀGINA -->

<HTML>

<BODY>
<!-- COS DE LA PÀGINA -->
...
<!-- FI COS DE LA PÀGINA -->
</BODY>

</HTML>
```

Codi Font 8.1 – Esquelet Mínim d'una Pàgina Web

8.3.1.1 Preamble de Pàgina

Com hem comentat, el preamble de la pàgina ha de ser el responsable d'obtenir tota la informació necessària de l'entorn de l'aplicació web per tal que la pàgina puga ser construïda correctament.

Imaginem el següent escenari: suposem que estem dissenyant una pàgina web (de venta de llibres) en la que es recuperaran tots els llibres de base de dades i per a cada llibre els seus autors. Volem, a més a més, que l'usuari puga seleccionar un autor i navegar a una altra pàgina en la que es mostre informació detallada d'aquest autor.

Si pensem en com deuriem implementar aquest comportament, ens adonarem ràpidament que fa falta que la pàgina llibre proporcione un enllaç associat a cada autor, de manera que aquest enllaç ens redirigisca a la pàgina *Autor_Detalls* i a més a més li "passe" l'identificador de l'autor seleccionat.

Per la seua part, la pàgina *Autor_Detalls* deurà accedir a la base de dades per recuperar sols l'autor que tinga com identificador aquell passat com argument en la navegació.

És per açò que el PREAMBLE de la pàgina web deu precedir a la resta de la pàgina, ja que la resta de la pàgina pot tenir una dependència de dades amb la informació del preamble.

Hi ha 3 accions a realitzar dins el preamble d'una pàgina web: (1) recollir els arguments d'entrada de la navegació, (2) provocar el canvi navegacional i (3) restringir/controlar l'accés dels usuaris.

Arguments d'Entrada. Una de les primeres accions que es deu realitzar dins el preamble és comunicar-se amb l'entorn de l'aplicació. Aquesta comunicació es realitza a través dels arguments navegacionals que viatgen fins la pàgina.

Els arguments que s'estan tenint en compte per fer el desenvolupament actual d'aplicacions en OOWS són els següents:

- **Pàgina d'Origen** de la navegació. Indica la pàgina de la que

Figura 8.19 – Exemple de navegació per seqüència en Amazon.com

venim per arribar a l'actual.

- **Camí Navegacional**. Indica el conjunts de *passos* que hem donat per arribar fins la pàgina actual.

- **Mode de Navegació**. Segons s'ha establer al model conceptual web, existeixen dues maneres d'accedir a un node navegacional: per exploració o per seqüència. En la navegació per *exploració* (causat per un enllaç d'exploració) i no duem informació contextual. La pàgina destí ha de recuperar tota la població de la base de dades.

En la navegació per *seqüència* (causat per seguir una relació navegacional de contexte), duem informació contextual al destí. La pàgina destí haurà de recuperar només la informació de la base de dades relacionada amb la informació contextual.

- **Informació contextual**. Quan la navegació siga per seqüència, produïda per haver seleccionat l'àncora d'una relació de contexte (un autor, per exemple), arribarà a la pàgina destí l'identificador de l'objecte seleccionat.

- **Usuari conectat**. En el cas de pàgines web que necessiten una identificació d'usuari, necesitarem saber quin usuari és el que està emprant l'aplicació.

Per tant, la primera secció del preamble de la pàgina deurà encarregar-se de la gestió dels arguments d'entrada, deixant-los accessibles per a que la resta de la pàgina (preamble i cós) puga emprar-los.

L'esquelet de la pàgina, introduint aquesta gestió d'arguments quedaria de la següent manera:

```
<!-- PREAMBLE DE LA PÀGINA -->

<!-- ARGUMENTS ENTRADA -->
...
<!-- FI ARGUMENTS ENTRADA -->

<!-- FI PREAMBLE DE LA PÀGINA -->

<HTML>

<BODY>
<!-- COS DE LA PÀGINA -->
...
<!-- FI COS DE LA PÀGINA -->
</BODY>

</HTML>
```

Codi Font 8.2 – Esquelet de Pàgina amb Arguments d'Entrada

Estat Navegacional. Es considera estat navegacional a les següents propietats:

- **Pàgina Actual**. Identifica el contexte/pàgina en la que està l'usuari actualment. S'utilitza per construir el camí navegacional.
- **Camí Navegacional** Seqüència de pàgines seguides per arribar fins la pàgina actual.

Al preamble de cada pàgina web deu provocar-se el canvi d'estat navegacional. És per açò que cada pàgina web deurà realitzar dues operacions amb respecte a aquest estat navegacional:

- establir l'identificador de la pàgina actual que serà emprat quan es navegue a una altra pàgina, sabent així la pàgina origen de la navegació
- recalcular el camí navegacional (rebut com argument), en funció del mode de navegació (exploració o seqüència). Així, si la navegació fins aquesta pàgina ha sigut:

  per seqüència, sabem que l'usuari ha seleccionat alguna informació en la pàgina origen i en aquesta pretèn veure la informació detallada. El camí navegacional deu crèixer afegint la pàgina actual. Aquest comportament és deu al fet que l'usuari està seguint una seqüència de navegació. La manera més habitual de construir un camí navegacional és crear una llista ordenada amb el nom de les pàgines separades per algun símbol (com ara "$>$" ó "::"). Per exemple: *Llibress* $>$ *Detalls d'Autor.*

  per exploració, l'usuari manfesta un canvi en la seqüència d'accions que estava realitzant i ara selecciona un enllaç del menú navegacional per començar a realitzar altre tipus d'operació.

El menú navegacional deu reiniciar-se i establir-se únicament a la pàgina actual.

Després de recollir els arguments d'entrada, podem provocar el canvi navegacional. Aquesta dependència es deu a que per a realitzar el canvi d'estat és necessària informació que s'arreplega en el preamble, com per exemple el mode de navegació i el contexte/pàgina d'origen.

```
<!-- PREAMBLE DE LA PÀGINA -->

<!-- ARGUMENTS ENTRADA -->
<!-- FI ARGUMENTS ENTRADA -->

<!-- ESTAT NAVEGACIONAL -->
<!-- FI ESTAT NAVEGACIONAL -->

<!-- FI PREAMBLE DE LA PÀGINA -->

<HTML>

<BODY>
<!-- COS DE LA PÀGINA -->
...
<!-- FI COS DE LA PÀGINA -->
</BODY>

</HTML>
```

Codi Font 8.3 – Esquelet de Pàgina incloent canvi d'estat navegacional

Identificació d'Usuaris. Quan es tracte de pàgines web que requerisquen d'un usuari registrat (aquelles pàgines que són la implementació d'un mapa navegacional que pertany a un usuari registrat), en el preamble deurem ficar codi que comprove que efectivament existeix un usuari identificat. En cas contrari, el sistema ens deurà reconduir a una pàgina d'informació on s'indique que cal identificar-se prèviament a accedir a aquesta pàgina.

Aquesta comprovació és necessària a causa que un usuari sense identificació podria intentar navegar a una pàgina, pel seu compte, que pertany a algun usuari registrat, sense haver-se identificat. La pàgina

ha d'estar doncs preparada per a açò, detectant aquest fet i redirigint-lo fins una pàgina d'error.

8.3.1.2 Cós de la Pàgina

El cós de la pàgina és la zona de la pàgina web que defineix els continguts. Aquestos continguts estan en funció del tipus de pàgina, com s'ha vist a la Taula 8.1. A causa d'açò, cada tipus de pàgina tindrà una implementació diferent del seu cós.

Seguint l'estratègia proposta de construcció de pàgines a partir de la composició de zones de contingut, la manera d'abordar com implementar la capa d'interfície se simplifica.

Una vegada definits els tipus de pàgines més habitual i els continguts que aquestes pàgines deuen (o poden) tenir, implementar-los resulta relativament senzill: devem introduir en la secció del cós de la pàgina tantes zones de contingut com siga necessari.

```
<!-- PREAMBLE DE LA PÀGINA -->
...
<!-- FI PREAMBLE DE LA PÀGINA -->

<HTML>

<BODY>
<!-- COS DE LA PÀGINA -->

<!-- ZONA 1 -->
...
<!-- FI ZONA 1 -->

<!-- ZONA 2 -->
...
<!-- FI ZONA 2 -->


<!-- FI COS DE LA PÀGINA -->
</BODY>

</HTML>
```

Codi Font 8.4 – Esquelet de Pàgina amb cos

Si analitzem un moment aquest estratègia de desenvolupament i la comparem amb les estratègies de desenvolupament habitual web, podem observar els següents avantatges:

- El desenvolupament de cada pàgina web que va a constituir l'aplicació web es realitza tenint en compte consideracions globals de l'aplicació (comunicació entre pàgines, homogeneitat, etc.), però de manera que cada pàgina és un ent relativament independent i configurable de la implementació de la resta de pàgines.

- Existeix definida una estructura conceptual per cada pàgina web, cosa que permet explicar clarament els objectius i responsabilitats de cada contingut que apareix a les pàgines implementades.

- Hi ha un estudi de diferents tipus de pàgines (en funció dels objectius de les pàgines) i de continguts, que permet pensar en un desenvolupament basat en l'"ensamblament" de components , més que en un desenvolupament més clàssic. Aquest enfoc ens proporciona certa flexibilitat en el desenvolupament, ja que per al disseny final de les pàgines podem anar afegint i/o llevant les zones de contingut que ens interesse.

- Inherentement, per l'aplicació d'aquest marc de treball, estem introduint elements que milloren la qualitat dels prototipus/aplicacions desenvolupades, ja que obliga a pensar en com deuen ser les pàgines web i quins continguts deuen proporcionar en funció del seu tipus.

- A més a més, aquesta estratègia ens pot servir per analitzar les aplicacions web ya desenvolupades i detectar virtuts i carències, inclús servir d'estratègia per al possible manteniment de les pàgines..

8.4 Introducció d'Aspectes de Disseny Gràfic

No hi ha cap dubte que la *Capa d'Interfície* no es limita únicament en mostrar informació. Cada vegada cobra més rellevància que aquesta poseisca un disseny impactant però a la vegada funcional. Es pot observar com la Web va enriquint-se mostrant dissenys cada vegada més espectaculars basats en animacions o generades amb eines de retoc professional. Per tant, no cap el menor dubte que encara que la finalitat d'una aplicació Web siga la de captar l'atenció de potencials clients, els requeriments visuals per aconseguir-ho són un fet.

La tasca de definir l'aspecte que les aplicacions Web ha de tenir, no recau (o no deuria recaure) als mateixos programadors, sinó en dissenadors gràfics. El que sí és cert, és que aquest grup professional normalment no tenen nocions sobre desenvolupament de programari (ni de llenguatges de programació, eines, tecnologies, etc.), i es centren en triar tipografies, estructures de visualització, colors, etc. Tanmateix, de la mateixa manera que els requeriments no funcionals de l'aplicació, aquestos requeriments de disseny són també cambiants i les aplicacions Web actualment no estan dissenyades per ser massa flexibles davant aquestos canvis.

En aquesta secció es presenta el problema de com abordar la definició de l'aspecte visual de les aplicacions Web generades mitjançant l'entorn de desenvolupament que es presenta a la tesi. Actualment, aquestos detalls o requeriments visuals no són contemplats en cap lloc dels models conceptuals proposats. La principal raó és que aquestos aspectes visuals no són definits per l'analista, sinó per un dissenyador gràfic.

8.4.1 Situació actual. Problemàtica i Objectius

Abans de descriure el problema, cal reflexionar una mica en quin seria l'entorn de treball adequat per definir els aspectes visuals d'una aplicació

Web. Els *dissenyadors* necessiten normalment llibertat a l'hora d'establir l'aspecte d'una aplicació Web, seguint unes pautes inicials normalment relacionades amb la creació d'"image corpora", estratègies de màrketing, atracció d'usuaris, etc. I per a fer-ho, tenen les seues d'eines centrades en el disseny visual, en la línia de l'Adobe Photoshop, que no introdeixen en cap moment aspectes de programació d'aplicacions.

D'altra banda, els *programadors* sols haurien de preocupar-se d'implementar l'aplicació Web, de manera que satisfaguen els requeriments inicials del sistema, i no barrejar els aspectes de disseny visual. És a dir, pareix clar que disseny i funcionalitat haurien d'estar separats, però açò no és realment un fet actualment, i no és un problema trivial.

Quan sorgí HTML, el seu objectiu principal era ser capaç de definir documents acadèmics que serien distribuits a través de la WWW. Als seus inicis importava pocs la tipografia o estructura del text. Tanmateix, amb l'augment creixent d'Intenet, prompte s'aplicà a un ambient més general, convertint-se en un nou mitjà de comunicació habitual. És en aquest moment quan no sols s'ha de compartir informació, sinó que aquesta deu ser atractiva de cara a l'usuari.

En les versions estándars posteriors a HTML, es feu un treball fort en dotar a aquest d'elements que permeteren estructurar la informació i definir com aquesta s'hauria de visualitzar. A partir del llenguatge HTML 3.0, estadaritzat per la W3C[1], quedà definida aquesta estratègia de barrejar contingut amb mode de visualització.

L'evolució de la Web ha conduït a la introducció de llenguatges addicionals a HTML, que introdueixen aspectes de comportament a la informació. Amb aquesta evolució, la gran majoria de les pàgines Web es formen fonamentalment amb dos llenguatges: 1) HTML per a definir els aspectes de contingut de dades i presentació d'aquestes; i 2) llenguatges dinàmics o d'*script* que poden executar-se tant en la

[1]World Wide Web Consortium

part client (JavaScript, Perl, VBScript o Applets, entre altres) com en la servidora (CGI, ASP, PHP o Servlets, entre altres). En qualsevol cas, el codi està inserit dins el propi HTML, de manera que el treball en la construcció del disseny visual i la funcionalitat estan lligats. En poques paraules, tant el dissenyador com el programador treballen sobre els mateixos recursos.

Si a aquest problema afegim la inherent característica de les aplicacions Web de "canviar" els seus requeriments, implica que les tasques de manteniment i evolució del programari es compliquen, forçant la interacció continuada entre dissenyadors i programadors. És per açò que cada vegada està més clara la necessitat del tàndem *dissenyador-programador* en el desenvolupament Web, per tal de tenir una visió més global del procés.

Els frameworks basats en el patró *Model-Vista-Controlador* o tecnologies com la recent ASP .NET introdueixen a aquest aspecte una novetat important, separant el codi dinàmic del HTML. A través de *vistes* es defineix el codi HTML estàtic que es completa mitjançant crides al controlador depenent de la informació que desitge l'usuari. L'aparença visual s'implementa amb aquesta aproximació s'implementa en aquestes *vistes HTML*, mentre que el codi de l'aplicació és implementat en les classes *controlador*.

D'aquesta manera, el dissenyador pot definir una vista amb certa independència, sense afectar a la funcionalitat. D'altra banda, el programador pot reutilitzar el codi d'una manera més eficaç gràcies a tots els mecanismes que li ofereix la programació orientada a objectes. Únicament haurà de marcar els controls a emprar i a partir d'ahí implementar el codi corresponent.

Però, aquest sistema no resol el problema completament, ja que sorgeixen incompatibilitats entre el disseny i la funcionalitat. Tanmateix, proporciona un mecanisme base per al desenvolupament, aplicable a la

Web, més flexible que la incrustació de tot el codi en HTML. Aquest últim cas, se'l sol anomenar com "codi spaghetti".

En esència, la solució ideal no és altra que la capacitat de definir de manera separada els aspectes de visualització. Aquesta idea dugué a la creació de *plantilles* que contenen la descripció dels aspectes visuals de les pàgines d'una aplicació Web. Tanmateix, aquest sistema força a l'establiment d'alguna manera d'unes *marques* al codi HTML, que després són referenciats des de les plantilles. Aquestes marques no estan predefinides, podent-se definir qualsevol conjunt, bé pels programadors, bé pels dissenyadors. Qualsevol modificació a aquestes marques pot provocar canvis en la feina d'ambdues branques.

A banda d'aquesta idea de treball, en principi encertada, apareix un greu problema: la reutilització del disseny. Les aplicacions Web solen tenir determinats elements comuns entre les seues pàgines, com ara les capçaleres, menús, estils de lletres, etc. Pareix lògic que allò ideal fora definir aquestos elements comuns i que aquestos foren heretats per les pàgines de l'aplicació. D'aquesta manera es podria moficar allò que es considerara necessari i els canvis es propagaren a tot el lloc web. Una cosa tan habitual encara no està suportada directament per cap tecnologia de desenvolupament Web.

Com solució general al problema de l'aparença visual de les aplicacions Web, el W3C proposà l'adopció dels *fulls d'estils en cascada* ("*Cascade Style Sheets, CSS*") [141]. Bàsicament, un full d'estils permet definir d'una manera organitzada regles d'estils (com la tipografia, colors, espaiats, disposició, ...) aplicats a les etiquetes d'HTML. Les regles CSS es poden definir aplicats directament als propis elements o bé fer ús de marcat especial (definint identificadors especials creats pel desenvolupador) i aplicar-ho a aquest marcat, independentment de l'etiqueta HTML.

Els CSS solen estar definits en fitxers separats i compartits per totes les pàgines d'una aplicació Web, de manera que els elements comuns

comparteixen estils de manera natural. Els avantatges són evidents. D'una banda podem modificar tot l'aspecte d'una pàgina, d'una altra banda, el codi estarà "net" de regles de visualització.

L'ús de CSS aporta gran flexibilitat en l'aspecte de la personalització de l'aparença. Tanmateix, es deu tenir en compte com classificar els elements. Devem establir les zones que deuen tenir una aparença comú en la nostra aplicació i definir el marcat (identificadors) correctament per que posteriorment es puguen aplicar aquestes plantilles CSS. Així doncs, els elements que formen part del menú seran inclosos en sota una identificació, la capçalera sota altra, i així amb la resta de zones comunes.

Arribats a aquest punt en el que pareix que CSS resol completament el problema, podem ressenyar un problema que no resol adientment. Si bé és possible estructurar una aplicació Web de manera que es puga marcar amb identificadors compartits dins l'aplicació per establir les regles de visualització als fitxers CSS, seria molt interessant el que aquestes regles de visualització pogueren ser aplicades directament a altres aplicacions Web. El problema arriba quan per aplicar les regles, l'estratègia de marcat és decissiva. Ja que hi ha llibertat a l'hora de definir aquest marcat i no hi ha un estàndar o convenció d'identificadors a utilitzar, cada aplicació té definits els seus i és pràcticament impossible que dos aplicacions compartisquen aquest marcat.

Açò provoca que realment, les regles de visualització no siguen realment reutilitzables de manera general. De fet, reestructurar l'estratègia de marcat d'una aplicació ja implementada per que siga equivalent a una altra és una feina complexa, que requereix de vegades de reenginyeria a partir de la implementació de les pàgines web i també costosa.

Una solució molt interessant a aquest aspecte seria que existira un estàndar que servira per construir qualsevol aplicació Web. Si açò esdevinguera el problema es veuria reduït en complexitat dràsticament. Els navegadors web podrien tenir regles de visualització predefinides i l'usu-

ari canviar-les dinàmicament, inclús podent importar i exportar aquestes regles entre aplicacions, existint repositoris on-line de visualitzacions. A les postres, ajudaria a simplificar el problema i proporcionar un marc comú de treball.

És a dir, els fulls d'estils en cascada són un bon mitjà per abordar la solució el problema de la introducció d'aspectes de disseny visual, però no són en sí la solució al problema.

Per tractar de donar una solució concreta a aquestos problemes, l'estratègia d'implementació d'OOWS respecte al disseny visual de les aplicacions Web es basa en els següents principis:

- *Separació de Conceptes*: els *analistes informàtics* han de poder definir els sistema d'informació sense tenir en compte cap aspecte visual, ni definir cap tipus de codi associat. D'altra banda, els *dissenyadors gràfics* han de definir l'aparença visual en base a unes directrius que no estan relacionades amb la tecnologia d'implementació del sistema. Però, la tasca realitzada per ambdós ha de poder fusionar-se d'una manera sistemàtica i controlada.

- *Reusabilitat*: Un disseny gràfic deu poder ser emprat tant dins d'una mateixa aplicació com per diferents aplicacions Web d'una manera natural i senzilla.

- *Adaptativitat*: El disseny gràfic deu poder ser adaptat a les peculiaritats d'un domini concret. És a dir, a partir d'un disseny visual genèric, adaptar-lo a necessitats concretes d'una aplicació Web concreta, aplicant modificacions que enriquisquen l'aparença visual.

- *Patrons de Visualització*: En la Web poden detectar-se patrons a nivell d'interfície que són emprats repetidament per diferents aplicacions. Aquestos patrons de visualització, fins cert punt

podrien considerar-se "estàndars" i haurien de poder incorporar-se al disseny d'una aplicació. Patrons com ara disposició bàsica d'elements (en columna, o "les opcions de navegació a l'esquerra i el contingut a la dreta", ...), o com visualitzar les opcions de navegació (com un llistat d'enllaços, com una paleta de pestanyes, ...), podrien ajudar a una definció ràpida del disseny visual de l'aplicació, agrupant per components d'una granularitat major.

L'estratègia que OOWS empra per aplicar aquestos principis és la següent:

1. Emprar llenguatges específics per a la definició de les regles de visualització (CSS) i fer-ho en fitxers separats: cap de les pàgines implementades proporcionarà informació de les regles de visualització. A més a més, s'emprarà XHTML [142] enlloc d'HTML. XHTML és el subconjunt d'etiquetes d'HTML encarregades de definir contingut, i que no defineixen com aquest contingut s'ha de visualitzar.

2. Definir un marcat de les pàgines web basat en termes conceptuals i no en termes d'implementació. Açò li dota a l'estratègia de major reusabilitat, ja que es basa en emprar les mateixes primitives que descriuen qualsevol aplicació web.

3. Crear dos fitxers per definir les regles de visualització: (1) un fitxer de regles *dependents de domini*, que inclouen els requeriments de l'aplicació actual (o del mateix domini) i empraran termes concrets d'aquest domini, i (2) un fitxer de regles *independents del domini*, que incloguen les regles genèriques que seran aplicades a qualsevol pàgina web, de manera independent al domini d'aplicació.

4. Publicar el repositori de dissenys gràfics i els patrons de visualització. Aquest repositori podria ser emprat per les pàgines web

implementades per obtenir les regles de visualització.

La implementació del Framework d'Interfícies Web (Secció 8.5.1) és responsable d'introduir el marcat adient a les pàgines Web, i enllaçar els dos conjunts de regles de visualització (dependents i independents de domini). La Secció 8.4.2 explica com definir aquest marcat i com aplicar-lo a la definició de les pàgines Web.

8.4.2 Estratègia de Marcat

Per complir amb aquestos requeriments, en l'entorn de desenvolupament definit s'ha apostat per una estratègia per definir els aspectes estètics ("*look and feel*") basada en *plantilles de presentació* definides mitjançant el llenguatge CSS. Aquest llenguatge permet definir les propietats visuals dels elements HTML d'una manera separada al codi de la pàgina. Aquesta característica és la que permet dur a terme la separació de rols, ja que el dissenyador gràfic pot encarregar-se de treballar en aquesta plantilla, i no dependre dels fitxers HTML generats per l'entorn. A més a més, evita que per error, el dissenyador puga canviar codi d'aplicació, introduint errors d'implementació.

Les plantilles CSS estan formades per regles que defineixen el conjunt de propietats visuals, no sols associades a etiquetes HTML (com ara <A> o <DIV>), sinò que és possible definir els nostres elements de marcat emprant la propietat *id* o *class* de CSS, que creen marques d'usuari al codi HTML.

El codi HTML que genera el Framework WIF (Secció 8.5.1) té associat un conjunt d'etiquetes, que fan referència als constructors conceptuals que provenen dels Models Conceptuals Web d'OOWS/OO-Method (Model d'Estructura d'Informació, Model Navegacional, etc.) sobre el que el framework es substenta.

Podem distingir dos tipus clarament d'etiquetes, que estableixen dos plantilles complementàries de regles de visualització:

- Les regles **Independients del Domini** es defineixen a partir de les primitives conceptuals d'OOWS/OO-Method, i els termes que introdueix el framework. Aquest tipus de regles fan referència a etiquetes com ara: *InformationZone* (zona d'informació), *AttributeName* (nom d'atribut), *NavigationLink* (enllaç navegacional), etc. Com es pot observar, aquestos conceptes són genèrics i poden emprar-se a qualsevol aplicació Web, siga del domini que siga. Aquest és el primer pas per aconseguir **reusabilitat** entre aplicacions.

- Les regles **Dependents del Domini** es defineix a partir de conceptes propis del domini de l'aplicació a construir, és a dir, a partir dels seu Model Conceptual. Suposant l'exemple de IMDb (Annexe C), per exemple, les marques estarien basades en conceptes com: *Movie*, *Director*, *GenreName*, etc. Aquestos conceptes són específics d'aquest domini d'aplicació, i per tant no són aplicables en general a altres aplicacions. Però, d'altra banda, són la base per **adaptar** una plantilla genèrica (obtesa a partir de regles Independents del Domini) i adaptar-la als requeriments específics d'aquest domini.

Ambdues plantilles són complementàries ja que, un element HTML de la interfície pot tenir associades més d'una regla de visualització en CSS. Algunes serien depenents del domini, i altres independents. En cas de conflicte, ha de prevaleixer la dependent del domini, per ser més específica.

La Figura 8.20 a la pàgina següent mostra un fragment de codi HTML produït pel framework, en el que s'aprecia el marcat d'ambdós tipus d'etiques. Com es pot veure, la divisió principal <DIV> constitueix una zona d'informació, i va marcada amb les etiquetes: 1) "MovieInformation" (dependent del domini), i 2) "InformationZone" (independent de domini).

```
<div class="MovieInformation InformationZone" >
    <table class="InformationView">
    <tr>
        <td class="attributeName MovieTitle">Title</td>
        <td class="attributeValue MovieTitle">
            <span class="TextValue">The GodFather</span>
        </td>
    </tr>
</div>
```

Figura 8.20 – Exemple de codi HTML produït pel Framework

	Etiquetes Independ.Domini	**Etiquetes Depend.Domini**
Cós Pàg. Web	*Context*, *Subsystem*	Nom del contexte o subsistema
Zona Informació	*InformationZone*, *AIU*, *ManagerClass*, *ComplementaryClass*, *AttributeName*, *AttributeValue*, *OperationName*, ...	Nom de la *Zona*, de l'*AIU* les classes navegacionals, atributs, operacions, ...
Zona Navegació	*NavigationZone*, *NavigationLink*, *NavigationGroup*	Nom dels contextes i subsistemes de cada *PahtStep*
Zona Ubicació	*LocationZone*, *Path*, *PathStep*	Nom dels contextes o subsist. de cada *PahtStep*

Taula 8.2 – Principals Etiquetes de Marcat proposades per OOWS

La Taula 8.2 mostra les zones web més representatives, i les etiquetes *Dependents de Domini* (DD) i *Independents de Domini* (DI) que s'empren en aquestes zones.

8.4.3 Regles de Visualització

Les regles de visualització defineixen com han de ser visualitzats els elements d'una pàgina web, amb respecte a la seua ubicació, tamany, color, tipus de text, etc, depenent del tipus d'element al que l'etiqueta vaja associada. Aquestes regles es defineixen en fitxers addicionals emprant el llenguatge CSS. La Figura 8.21 a la pàgina següent mostra un exemple de la pàgina web que genera el framework, incloent el marcat tant dependent com independent del domini.

Com s'ha comentat a la Secció 8.4.2, l'estratègia proposada defineix dos tipus de regles de visualització: les *dependents del domini* i les *independents del domini*. Per tant, hi haurà dos plantilles (repartides en fitxers diferents) amb aquestos grups de regles.

Combinant aquestes idees amb l'estratègia de marcat i l'ús del llenguatge CSS, poden definir *regles de visualització* realment complexes. Podem inclús definir regles que involucren aspectes dependents com independents del domini. Si una regla es defineix d'aquesta manera, haurà de considerar-se com dependent del domini, ja que empra etiquetes d'aquest tipus que no seran reutilitzables a nivell genèric.

Per tant, els dissenyadors gràfics poden enfocar els seus esforços en definir aquestos fitxers. No hi ha ja necessitat d'interactuar amb els desenvolupadors de l'aplicació, ja que coneixen a priori, a partir dels models i de l'estratègia de marcat, les marques que tindrà l'aplicació final. D'alguna manera, aquesta estratègia és la que serveix com *contracte* entre les tasques de disseny visual i les tasques d'implementació, sentant les bases per separar els esforços i combinar-los d'una manera senzilla.

Vegem els següents exemples de regles de visualització, aplicat al cas d'estudi IMDb (Annexe C):

- *Requeriment de Visualització*: "Ubicar la zona de navegació de cada pàgina web a la part superior de la pàgina. Separar els enllaços navegacionals d'aquesta zona mitjançant una barra vertical. Totes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title></title>
<link rel="stylesheet" type="text/css" href="style/dd-IMDb.css" />
<link rel="stylesheet" type="text/css" href="http://ascalon.dsic.upv.es/Styles/IMDb/di-IMDb.css" />
</head>

<body class="Context" id="NC_Movie_Overview_MainDetails">

<div class="LocationZone">
 <div class="Path">
    <div class="PathStep"><a class="PathStep" id="Movie_Overview_MainDetails" href="MainDetails.php"><span class="PathStep"
id="NS_Movie">Movie</span></a></div>
    <div class="PathStepSeparator"><span class="PathStepSeparator"></span></div>
    <div class="PathStep"><a class="PathStep" id="Movie_Overview_MainDetails" href="MainDetails.php"><span class="PathStep"
id="NS_Overview">Overview</span></a></div>
    <div class="PathStepSeparator"><span class="PathStepSeparator"></span></div>
    <div class="PathStep"><a class="PathStep" id="Movie_Overview_MainDetails" href="MainDetails.php"><span class="PathStep"
id="NC_MainDetails">main details</span></a></div>
 </div>
</div>


<div class="NavigationZone">
<div class="NavigationGroup">
 <div class="NavigationLink" id="NC_NowPlaying"><a class="NavigationLink" id="NC_NowPlaying" href="NowPlaying.php"><span
class="NavigationLink" id="NC_NowPlaying">NOW PLAYING</span></a></div>
<div class="NavigationLink" id="NC_MovieNews"><a class="NavigationLink" id="NC_MovieNews" href="MovieNews.php"><span
class="NavigationLink" id="NC_MovieNews">MOVIE/TV NEWS</a></div>
<div class="NavigationLink" id="NC_MyMovies"><a class="NavigationLink" id="NC_MovieNews" href="MY Movies.php"><span
class="NavigationLink" id="NC_MovieNews">MY MOVIES</a></div>
BOARDS</a></div>
...
</div>
</div>

<div class="InformationZone">

<div class="AIU" id="AIU_Movie_Main_Details">
<table class="ManagerClass" id="Class_Movie">
 <tr><th class="AttributeName" id="Class_Movie_Title"><span class="AttributeName" id="Class_Movie_Title">Title</span></th>
     <td class="AttributeValue" id="Class_Movie_Title"><span class="AttributeName" id="Class_Movie_Title">The
Godfather</span></td></tr>
 <tr><th class="AttributeName" id="Class_Movie_Year"><span class="AttributeName" id="Class_Movie_Year">Year</span></th>
     <td class="AttributeValue" id="Class_Movie_Year"><span class="AttributeName"
id="Class_Movie_Year">1972</span></td></tr>

...
</table>
</div>

<div class="AIU" id="AIU_User_Comments">
...
</div>

</div>


</div>
</div>

...

</body>
</html>
```

Figura 8.21 – Pàgina Web amb Marcat

les pàgines deuen mostrar el "*logo*" de IMDb". Aquestos són 3 exemples de regles de visualització independents de domini, ja que no estan definides en termes del model conceptual d'IMDb, i podria extrapolar-se a qualsevol aplicació Web..

```
...

.Context {
        background-image:url("logo.jpg");
}

.NavigationZone{
        z-index:1;
        position: absolute; top:35px; left: 170px;
        font-weight:bold;
        font-size:9px;
}

NavigationGroup div.NavigationLink + div.NavigationLink{
        border-left-style:solid;
        border-width:thin;
        padding-left:5px;
}

...
```

Figura 8.22 – Exemple de Marcat Independent de Domini

- *Requeriment de Visualitzacio*: "Els títols de les pel·lícules (*Movie*) deuen aparèixer en lletra gran, negreta". Aquest és un exemple de regla de visualització depenent del domini, ja que sols es pot aplicar al domini actual de l'aplicació, on hi ha informació de tipus *Movie* que té la propietat *títol*.

Per satisfer els requeriments de visualització independents de domini presentats, la Figura 8.22 mostra un exemple de codi en CSS que ho representa. Com podem apreciar, s'especifica que tots els contextes han de tenir com a imatge base un *logo.jpg* (que serà el de IMDb). Les zones de navegació s'han de possicionar de manera absoluta a la part superior de la pàgina (top=35px deixa un marge petit de la part superior). Finalment, quan es troben dos enllaços de navegació seguits dins una zona de *NavigationGroup* (que defineix un menú de navegació al framework), s'estableix un border (creant una línia) i una separació de 5 píxels entre els enllaços.

D'altra banda, la Figura 8.23 a la pàgina següent mostra la regla de

```
...

#Class_Movie_Title {
        font-weight:bold;
        font-size:15px;
}


...
```

Figura 8.23 – Exemple de Marcat Dependent de Domini

visualització que diu que el tipus de lletra associat al títol (Title) de les pel·lícules (Class_Movie) han de mostrar-se en negreta (font-weight:bold) i un tamany de 15 pixels.

Com es pot observar, totes aquestes regles es poden definir de manera ortogonal a la construcció de les pàgines Web, aconseguint la separació de tasques entre el dissenyador gràfic i els programadors.

8.4.4 El Paper del Framework

El framework (WIF) (Secció 8.5.1) és l'encarregat d'associar les marques de les plantilles de presentació al codi que implementa l'aplicació Web. Per a fer-ho, l'estratègia que segueix és la següent:

- En primer lloc, els constructors del framework produeixen codi XHTML sense cap tipus de format predefinit. Tanmateix, aquest codi el marca mitjançant un conjunt d'etiquetes genèriques (les independents del domini), que permeten al dissenyador gràfic distingir els diferents components de disseny de les pàgines Web. D'altra banda, també introdueix el marcat dependent del domini a patir de la informació extreta dels models conceptuals. Ambdós marcats es realitzen de manera completament automàtica, sense intervenció de cap programador o dissenyador. Per evitar col·lisions

en els noms de les marques, tant dependents com independents de domini, s'ha aplicat una estratègia de nomenat que té en compte aquest fet.

- D'altra banda, el framework proporciona un mètode, DefaultStyle, que pertany a l'objecte Aplicació (Annexe A), i ens permet definir la plantilla de visualització que s'aplicarà. Aquestes plantilles hauran d'estar publicades en un repositori de plantilles de visualització, al que l'aplicació farà referència. És una tasca del dissenyador gràfic emplenar aquest repositori de plantilles, a partir de l'estratègia de marcat definida. El framework s'encarrega d'associar aquestes plantilles al codi XHTML a mesura que les pàgines Web són accedides en temps d'execució.

Suposem, sobre l'exemple d'IMDb, que s'ha especificat la següent configuració:

```
$Application->DefaultStyle(“IMDB”);
```

El Framework generarà les següents línies de codi XHTML cada vegada que l'usuari accedisca a una pàgina de l'aplicació Web i carregarà les regles de visualització:

```
<link rel=“stylesheet” type=“text/css” href=“X/IMDB/DD.css” >
<link rel=“stylesheet” type=“text/css” href=“X/IMDB/DI.css” >
```

sent X la URL on està publicat el repositori d'estils de visualització, organitzats per carpetes (nom de l'estil) i contenint les regles dependents (DD.css) i independents (DI.css).

Aquesta estratègia basada en plantilles dependents i independents de domini ens permet complir d'una manera clara dos dels principis objectiu:

Figura 8.24 – Pàgina Web amb dos estils visuals diferents

- *Adaptabilitat*: Canviar el "*look and feel*" d'una aplicació consisteix en canviar la crida al mètode DefaultStyle, passant-li una nova plantilla. La Figura 8.24 mostra un exemple d'una mateixa pàgina web després d'aplicar-li dos plantilles de visualització diferents.

- *Reusabilitat*: totes les aplicacions Web desenvolupades mitjançant la *OOWS Suite* estan implementades de manera que comparteixen el marcat independent de domini (el mateix per a totes). D'aquesta manera, les plantilles independents de domini de qualsevol altra aplicació poden ser associades a qualsevol aplicació Web implementada d'aquesta manera. Si aquesta proposta de marcatge fora acceptada com un estàndar, l'afirmació anterior podria convertir-se en: "qualsevol plantilla independent de domini és aplicable a qualsevol aplicació Web desenvolupada, i a l'inrevés".

8.5 El Procés de Generació de codi

Una vegada definits els models conceptuals que especifiquen l'aplicació Web, el següent pas és compilar aquestos models per produir el codi equivalent a l'aplicació. En l'actualitat existeixen dues possibles alternatives en l'àmbit de la MDA per realitzar aquest procés. En primer lloc, l'aproximació *elaboracionista* proposa la definició d'una transformació de PIM a PSM prèvia a la generació de codi. En aquest cas, el desenvolupador pot completar el PSM amb informació addicional relacionada amb la tecnologia destí, parametritzant la producció del codi final.

La segona aproximació, anomenada de *transformació automàtica*, defén que a partir d'un PIM s'obtinga directament el codi mitjançant un conjunt de regles de transformació. En aquesta alternativa, el PSM segueix present, implícitament, en les regles de transformació, però no pot ser parametritzat, com esdevé en l'altra aproximació.

A l'hora de decidir quina aproximació MDA és més adequada per al nostre procés de generació, cal preguntar-se si és realment necessari tenir un PSM. Dividir la generació de codi en dues etapes simplifica la complexitat de les regles de transformació, però provoca que el procés general siga més complexe. A més, cal tenir el PSM per a les plataformes a les que va a generar-se, que no resulta trivial, a causa de la complexitat de les tecnologies d'implementació existents.

En conseqüència, el bot entre el PIM i el PSM és massa complex a pesar de la potència dels llenguatges de transformació de models. Açò és degut a que els llenguatges de programació Web imposen una arquitectura de desenvolupament molt allunyada semànticament de la que es pot obtenir a partir dels models conceptuals d'OOWS. D'altra banda, aquesta aproximació implica la definició d'un segon conjunt de regles de transformació que produisquen el codi a partir del model.

Altre inconvenient de les transformacions PIM a PSM és el fet

que l'analista ha de tenir coneixements tecnològics sobre el PSM a emprar, per tal de poder modificar-lo i manipular-lo. El mètode OOWS persegueix just el contrari: que l'analista definisca l'aplicació Web de manera independent a la tecnologia.

Per tant, ja que definir i emprar un PSM com a pas intermedi en les transformacions no anava a proporcionar cap avantage real en el procés de desenvolupament, es decidí descartar el seu ús explícit.

Ara bé, si produir codi a partir d'un PSM és complexe, obviament a partir d'un PIM la dificultat augmenta. Cal aleshores definir un mecanisme que permeta simplificar les regles de transformació. La solució presa ha sigut la implementació d'un *Framework d'implementació d'Interfícies Web* (WIF) que proporcione primitives d'alt nivell per a la construcció de les aplicacions. La seua missió és facilitar la delegació dels problemes d'implementació a nivell d'implementació (i no de les regles de transformació).

Així, per exemple, l'autenticació d'usuari implica la definició de gran quantitat de codi que realitze totes les comprovacions oportunes en un llenguatge de programació concret. Tanmateix, és possible construir una classe que encapsule aquesta funcionalitat i que la expose a través d'un conjunt reduït de mètodes (*Identificar Usuari*, *Difinir Rol d'Usuari*, etc.). Les regles de transformació, enlloc de generar tot el codi subjacent, únicament genera codi per realitzar les crides aquesta classe que encapsula aquesta funcionalitat.

Emprant aquesta aproximació, les regles de transformció són significativament simplificades i la transformació PIM a codi (model-a-codi) també es veu simplificada. Aquest framework WIF segueix aquesta aproximació definint un conjunt de classes que exposen la funcionalitat típica de les interfícies Web.

La Secció 8.5.1 el framework WIF, fent un repàs als constructors que proporciona. Després, a la Secció 8.5.2 es revisen els aspectes tecnològics

per definir les regles de transformació, de manera que aquestes puguen representar-se i executar-se correctament. A la Secció 8.5.3 es mostren les regles de transformació aplicades al framework WIF com tecnologia destí. Finalment, la Secció 8.5.4 mostra com el codi resultant de la transformació de models OOWS es comunica amb la lògica de negoci generada per *OlivaNOVA*, donant lloc a una aplicació Web completament funcional.

8.5.1 Framework d'Interfícies Web

El principal objectiu del *Framework d'Interfícies Web, WIF* és simplificar les transformacions model-a-codi. Aquest framework ha sigut desenvolupat amb PHP (versió 5) donada la seua àmplia acceptació per construir frameworks Web.

L'Annexe A proporciona informació més detallada i completa sobre aquest annexe. A més a més, en [128] es podrà trobar tota la informació relativa a aquest, a més d'actualitzacions i millores que se li vagen fent.

Per definir aquest framework s'han agafat alguns principis sobre les Factories de Programari. Així, per exemple, les primitives d'alt nivell que abstrauen la implementació de les pàgines web es proposen com primitives de construcció dins el propi framework. Gràcies a aquest nivell d'abstracció, el "gap semàntic" entre els models PIM i el codi (del framework) es redueix dràsticament.

D'aquesta manera, les transformacions PIM-a-PSM es simplifiquen al màxim ja que existeix una relació directa entre els conceptes del PIM (model conceptual web) i les primitives del PSM (primitives del framework). Els avantatges i incovenients d'aquesta aproximació es comenten en [6]. Aquestes primitives són definides com un conjunt de classes que representen els conceptes comuns de les aplicacions Web, com ara: pàgina web, enllaç, menú de navegació, servei, etc.

Seguint aquesta aproximació, una aplicació Web pot ser definida com

un conjunt d'objectes que especifiquen el tipus de funcionalitat que han de proporcionar. Els objectes definits pel framework per construir una aplicació Web són: *Application*, *Page* i *Zone*.

L'objecte **Application**, únic en l'aplicació, conté informació global de l'aplicació. Així, el mètode *Role* permet definir els diferents tipus d'usuaris que poden accedir a l'aplicació, mentre que la propietat *AllowAnonymous* habilita l'accés a usuaris anònims. També, sobre aquest objecte *Application* es defineixen el conjunt de pàgines amb el mètode *AddPage* (aplicat sobre objectes de tipus *Page*

L'objecte **Page** està relacionat amb la implementació d'una pàgina Web. Seguint els principis vistos en Secció 8.2, aquest objecte defineix una pàgina web com una agregació de zones de contingut. Una zona pot ser definida com una peça independent que composa la pàgina web, amb un objectiu clar. Aquest objecte *Page* proporciona mètodes per afegir aquestes zones de contingut, com ara *AddNavigationZone* o *AddInformationZone*.

L'objecte **Zone** defineix qualsevol contingut d'una pàgina. Per exemple, si definim una zona d'informació, aquest objecte proporciona el mètode *AddField* per seleccionar els atributs que es recuperaran ó el mètode *AddDetail* que mostra la informació recuperada a través d'una relació d'associació-agregació-composició o herència. Els mecanismes per definir *filtres* o *indexos* també s'especifiquen en aquest objecte mitjançant els mètodes *DefineFilter* i *DefineIndex*, respectivament.

Ja que aquestos conceptes no són únics d'OOWS (no provenen directament de les seues primitives), aquest framework podria ser emprat per altres mètodes d'enginyeria web en la seua fase de generació. A més a més, el framework facilita la inclusió d'aspectes de disseny estètic per mig de plantilles que poden ser adaptades i reutilitzades com es pot veure en [135].

La Figura 8.25 a la pàgina següent mostra un exemple de pàgina web

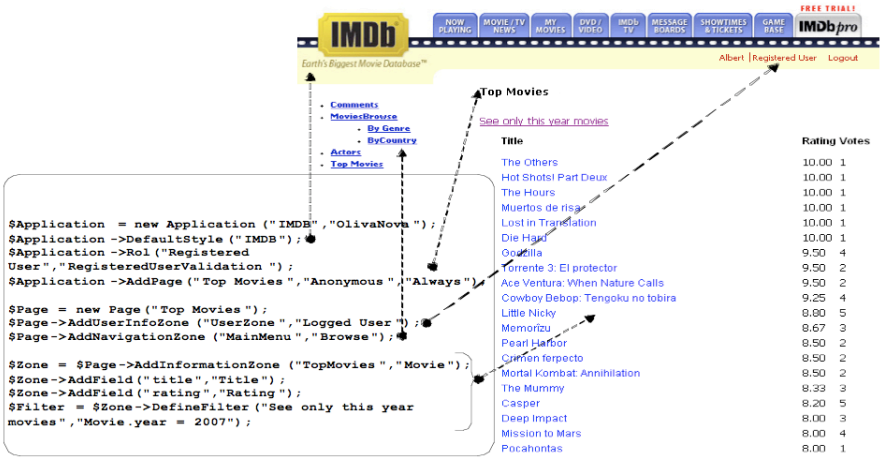


Figura 8.25 – Exemple de Codi Web emprant el framework WIF

del cas d'estudi de IMDb on es poden veure els objectes del framework WIF i una representació concreta. Quan un usuari fa una petició a l'aplicació, el framework crea un conjunt d'objectes que construeixen la pàgina web sol·licitada (*Application, Page, Zone*, etc.). Aquestos objectes produeixen una eixida en codi XHTML que es enviat al navegador client.

8.5.2 Regles de Transformació. Aspectes Tecnològics

Per definir les transformacions de model a codi, hi ha varies aproximacions. Alguns autors suggereixen l'ús de transformacions basades en grafs [69], altres mitjançant llenguatges de suport basades en plantilles [41], i altres que defensen l'ús de transformacions en XSLT [34].

D'aquestes, el primer de transformacions, les basades en grafs, són molt adients per realitzar transformacions model-a-model. Les regles defineixen l'homomorfisme entre els grafs. Però per a realitzar transfor-

macions model-a-codi ("*PIM-to-code*"), que és el que pretenem, no són massa adients.

Els altres dos grups de transformacions són en realitat basats en plantilles. XSLT és un exemple concret de llenguatge de plantilles, que empra com a base XML. No s'ha triat aquest tipus de transformacions per la limitació que imposen, ja que l'origen ha de ser necessàriament XML i la potència expressiva del llenguatge XSLT es queda curta en certes circumstàncies.

És per açò que hem seguit la filosofia de la transformació mitjançant plantilles, amb llenguatges específics creats a propòsit per a aquesta feina. *MOFScript* [43] i *openArchitectureWare Xpand* (oAW Xpand) [127], són dos llenguatges referència en aquest aspecte, i ambdós han sigut emprats en la definició de diferents generadors [128].

A l' Annexe D es mostren les transformacions emprant Xpand. El principal avantatge de Xpand amb respecte a altres llenguatges és que proporciona un bon suport dins l'entorn Eclipse [41], de manera que pot ser fàcilment integrat en l'entorn MDA proposat. Emprant aquest llenguatge, les regles de transformacions es defineixen a patir de les primitives conceptuals, que agafen com entrada. A partir d'aquesta informació conceptual es completa una plantilla de codi.

Per definir el procés de generació amb OOWS, s'han definit un conjunt de regles que agafen les primitives conceptuals d'OOWS com entrada. Cada element OOWS té una regla que produeix codi emprant directament codi del Framework d'Interfícies Web, sense pasar per un PSM intermig per realitzar les transformacions.

L'exemple de la Figura 8.26 a la pàgina següent il·lustra aquest procés. El Model Navegacional (descrit al Capítol 4) té una regla associada, anomenada ModelNavigationalRule que genera el codi que afig l'usuari com a *Rol* dins el framework (un usuari amb accés al sistema). Després, per a cada *Contexte Navegacional* que es trobe, es crea una

```
«DEFINE NavigationalModelRule FOR NavigationalModel»
    $Application->Rol("«UserRol.id»","«UserRol.class»","«UserRol.parent»");
«FOREACH Navigational Context AS NC»
    $Application->Page("«NC.id»","«NC.alias»","«UserRol.id»","«NC.reachability»");
    «EXPAND NavigationalContextRule FOR NC»
«ENDFOREACH»
«ENDDEFINE»
```

Figura 8.26 – Transformacions Model-a-Codi

pàgina en el framework amb la invocació a: $Application->Page().

Com es pot apreciar, les regles de transformació són prou simples gràcies al framework implementat, que redueix la distància dels conceptes del domini i de la solució, i internament dóna una solució a nivell d'implementació comuna a al conceptes del domini (que poden ser vists com patrons).

8.5.3 Patrons de Traducció emprant el Framework

Finalment, cal definir les transformacions entre les primitives de modelatge conceptual i el framework d'implementació. Aquest pas es duu a terme automàticament per algun dels compiladors de models que OOWS proporciona [128].

En essència, tots els compiladors es comporten de la mateixa manera: recorren tots els elements del metamodel d'OOWS (Annexe B) i executen les regles de transformació de cada element en base a la tecnologia destí emprada. Aquestes regles de transformació són més o menys complexes en base a la distància semàntica ("*gap*") entre els models i el codi. Aquest és un dels motius principals pels que s'ha definit el framework d'implementació d'interfícies web, que apropa els conceptes de modelatge i els d'implementació. L'Annexe D mostra en detall la implementació d'aquestes regles.

La primera regla de transformació s'aplica a l'element del metamodel

OOWSModel i és la que crea el projecte d'aplicació web, definint el nom de l'aplicació i la visualització per defecte en un fitxer de configuració d'aplicació[2].

A partir d'aquest element s'invoquen la resta de regles, com es descriu a continuació:

1. **Regles de Transformació per al Diagrama d'Usuaris**. Els usuaris es troben al metamodel a partir de *OOWSModel.UserRol*.

   (a) *Regla d'**usuari***: per cada usuari (*UserRol*) definit al model navegacional s'afig una operació AddRol al fitxer de configuració de l'aplicació. Per als tipus d'usuari *Anònim* no s'especifica operació de validació. Si un usuari és una especialització d'un altre usuari, s'indica a l'últim argument de l'operació AddRol.

2. **Regles de Transformació per al Mapa Navegacional (*Authoring-in-the-large*)**. El concepte de Mapa Navegacional no existeix explícitament al metamodel. Aquest mapa s'obté a partir de tots els nodes que té un usuari (*UserRol.NavigationalNode*).

   (a) *Regla per a **grups de pàgines***: per cada subsistema navegacional (*NavigationalSubsystem*) s'especifica una operació AddPageGroup al fitxer de configuració de l'aplicació. Aquest grup s'estableix com "sempre visible" i com a propietari l'usuari que s'està processant. Si està dins un altre subsistema, també s'indica amb aquesta operació.

   (b) *Regla per a **pàgines***: per cada contexte navegacional (*NavigationalContext*) de tipus exploració (*reachability='Exploration'*) s'afig una operació AddPage al fitxer de configuració d'aplicació, amb visibilitat establerta a "sempre

[2]Per a informació detallada sobre el framework, consultar l'Annexe A.

visible". Per cada contexte navegacional de tipus seqüència *reachability='Sequence'*) s'afig una operació AddPage amb visibilitat *fromPage*. Totes aquestes pàgines s'estableixen per a l'usuari propietari del mapa que s'està definint. Si un node pertany a un subsistema, s'indica a l'operació AddPage el grup al que pertany.

(c) *Regla per a la **pàgina per defecte** ("Home")*: un dels nodes navegacionals pot ser establert amb la propietat "Home", definint-lo mitjançant l'operació SetHomePage. Si no hi ha cap node marcat *per defecte*, el compilador de models crea una nova pàgina de primer nivell (AddFirstLevelPage) i marcant-la com pàgina per defecte amb l'operació SetHomePage. Cada subsistema navegacional també deu tenir la seua pàgina per defecte, i s'aplica els mateix algorisme per marcar-la, com si d'un mapa navegacional es tractara.

3. **Regles de Transformació per definir el Contingut dels Nodes Navegacionals (*Authoring-in-the-small*)**. Cada node navegacional ja ha sigut declarat per les regles 2a i 2b al fitxer de configuració de l'aplicació. En aquest punt, el procés de transformació crea el fitxer de definició de cada pàgina web, aplicant una de les següents regles.

   (a) *Regla per a definir un **contexte navegacional***: si el node navegacional és un contexte navegacional, es crea una pàgina web de tipus *informació* (Secció 8.2.1.1) que contindrà l'especificació d'aquest contexte. Per cada contingut d'informació (UAI) es s'especifica una nova zona d'informació mitjançant l'operació AddInformationZone, indicant la classe (*NavigationalClass*) a partir de la que es farà aquesta recuperació d'informació. Per cada atribut navegacional (*Navi-*

gationalAttribute) de la classe, s'especifica una AddField. Si la classe navegacional té operacions navegacionals (*NavigationalOperation*), es crea una (sub)zona mitjançant l'operació AddServicesZone, que incloura la referència a l'operació de la classe mitjançant l'operació AddReference[3]. Finalment, per a cada relació navegacional (*NavigationalRelationship*) s'especifica una operació AddDetailedRelationship, indicant la classe navegacional relacionada. Recursivament, aquesta classe navegacional "complementària" ha de ser processada, a partir dels seus atributs, operacions i relacions navegacionals.

(b) *Regla per definir un* ***subsistema navegacional***: per cada subsitema navegacional (AddPageGroup) es crea una pàgina de tipus "estructura de navegació" (Secció 8.2.1.2), que conté enllaços a totes les pàgines que estan dins el contexte i que han sigut creades mitjant la regla 2b, indicant que pertanyen a un grup.

4. **Regles de Transformació per definir els Mecanismes d'Accés**. Durant el processat de les zones de contingut (UAIs) d'un contexte navegacional, poden aparèixer especificats alguns mecanismes d'accés. Aquestes dues regles especifiquen les seues propietats.

   (a) Regla per definir un **índex**: per a cada index (*Index*) definit a la UAI s'especifica una nova zona dins la pàgina, DefineIndex, processant, si està especificada, la seua *vista de resultats*.

   (b) Regla per definir un **filtre**: per a cada filtre (*Filter*) definit a la UAI s'especifica una nova zona dins la pàgina, DefineS-

[3]Un dels passos previs a tot açò en el procés de generació és el de recòrrer les classes del diagrama de classes i crear les operacions. Per cada operació que es puga executar es registra al framework un Service que a més de contenir el formulari (pàgina de tipus "entrada de dades", Secció 8.2.1.3) per invocar-lo, manté la referència a la lògica d'aplicació que l'executarà finalment.

taticFilter ó DefineDynamicFilter (atenent al tipus de filtre), processant, si està especificada, la seua *vista de resultats*.

(c) Regla per definir una **vista de resultats**: per cada vista de resultats (*SearchView*) es defineix una (sub)zona d'informació associada al mecanisme d'accés processant el conjunt de visualització que defineix la vista (en forma de UAI), i indicant l'atribut de selecció (SelectionAttribute).

Per veure informació addicional de l'especificació completa i implementada en una tecnologia de definició de compilador *model-a-codi*, veure l'Annexe D.

8.5.4 Integració funcional amb la lògica de negoci

Com s'ha indicat al llarg d'aquest treball, a partir dels models OOWS es pot generar una interfície Web aplicant unes regles de transformació o compilació de models. Aquesta interfície Web no constitueix per sí una aplicació Web, ja que no proveeix la funcionalitat necessària. L'arquitectura del framework WIF ha sigut dissenyada de manera que fora allò més independent possible la implementació de la lògica de negoci. Així, és possible acoplar diferents lògiques de negoci a la interfície, o no lligar la interfície en funció de com estiga implementada la lògica.

Actualment, les implementacions del compiladors de models existents d'OOWS[4] s'han centrat en emprar la lògica de negoci generada per OO-Method/*OlivaNOVA*. Tanmateix, existeix la possibilitat tant de proporcionar una lògica de negoci implementada en PHP o mitjançant Serveis Web externs [116].

[4]En [128] poden trobar-se més compiladors: de OOWS a ASP, i OOWS a PHP, ambdós realitzats en MOFScript, disponibles en el moment de finalització d'aquesta tesi.

Per desacoplar la lògica de negoci, la solució adoptada ha sigut la d'aplicar el patró *façana de negoci* ("*facade*") de la colecció de patrons de [48]. L'objectiu d'aquest patró és el d'oferir una interfície única d'un subsistema per facilitar el seu ús. En el nostre cas, el subsistema està format pel conjunt de classes que proporcionen la funcionalitat. Emprant una única interfície predefinida, no sols simplifiquem les interaccions, sinó que per a la resta de classes del framework, és transparent com s'implemente aquest subsistema.

Per que les diferents lògiques de negoci definisquen la mateixa façana de negoci, s'ha definit al framework la interfície *Bussiness Facade*, que implementa un objecte façana concret. Aquesta interfície exposa els següents mètodes que hauran de ser definits, en funció de la lògica i tecnologia subjacent:

- *QueryPopulation*: retorna la població relacionada d'una classe determinada. Rep com argument l'identificador de la classe, una colecció amb els identificadors dels atributs a mostrar i un conjunt de filtres per restringir les instàncies a partir de les condicions de filtrar. Un exemple de consulta: "recuperar el nom i gènere (atributs) de totes les pel·lícules (classe) de l'any 2007 (filtre)".

- *QueryById*: retorna una única instància d'una classe a partir del seu identificador *oid*. Rep com arguments l'identificador de la classe, una colecció amb els identificadors dels atributs a mostrar i l'identificador únic de la instància a recuperar. Un exemple de consulta: "recuperar el nom i any (atributs) de la pel·lícula (classe) amb l'identificador (oid) de valor 115".

- *QueryRelated*: a partir de l'identificador *oid* d'una instància, retorna totes les instànces que pertanyen a altra classe relacionada. Rep com arguments l'identificador de la classe origen, una colecció amb els identificadors dels atributs a mostrar de la classe relacio-

nada, l'identificador únic (oid) de la instància de la classe origen, l'identificador de la relació entre ambdues classes i opcionalment una colecció de filtres. Un exemple de consulta: "recuperar els noms (atribut) de tots els actors (classe relacionada) de la pel·lícula (classe) amb identificador (oid) 115, que s'obté a través de la relació amb la classe actor anomenada Participa".

- *ExecuteService*: executa una operació que pertany a la lògica de negoci a partir de l'identificador únic d'un servei i el conjunt dels seus arguments. Com argument s'envia el valor (com una cadena de text) i el tipus de dades. El mètode retorna cert si l'execució del servei ha sigut correcta o fals en cas contrari.

- *GetServiceResponse*: recupera el resultat o l'error de l'últim servei executat en la façana de negoci. No rep arguments.

La façana de negoci descrita conté la funcionalitat mínima que necessita una interfície Web desenvolupada mitjançant el framework. La interfície presuposa que la lògica de negoci prové d'un diagrama de classes implícit. Els diferents mètodes retornen per tant els objectes i instàncies tal i com les entén el framework.

D'altra banda, la lògica d'integració ha de ser implementada manualment en funció de la plataforma destí. En el cas d'aplicacions generades amb *OlivaNOVA* o que seguisquen una arquitectura perfectament definida, únicament és necessari implementar la façana de negoci una sola vegada. Però, si la funcionalitat prové de Serveis Web dispars, aquesta aproximació no resulta possible.

Com avantatge, la lògica d'integració es troba definida en un únic lloc i totalment desacoplada de la resta del framework, de manera que l'impacte de canviar o introduir noves plataformes es minimitza. Per defecte, el framework incorpora una façana de negoci que es comunica amb aplicacions *OlivaNOVA*, que es descriu a continuació.

La lògica de negoci modelada amb *OlivaNOVA* es produeix a través d'un dels motors de transformació que proporciona. Aquestos motors, anomenats *OlivaNOVA Transformation Engines* [25], són implementacions de diferents processos de compilació de models conceptuals OO-Method. Cada motor implementa: (1) un *repositori del model conceptual*, (2) un *repositori del model d'aplicació*, (3) un *conjunt de correspondències entre els elements dels repositoris* anteriors i (4) un *conjunt de transformacions associades a cadascun dels elements del repositori del model d'aplicació.*

El repositori del Model Conceptual és comú a tots els motors de transformació i és capaç de carregar models emmagatzemats en XML d'intercanvi produïts per *OlivaNOVA Modeller*.

El resultat d'una transformació varia en funció de la tecnologia destí triada (.NET, J2EE). Tanmateix, l'arquitectura de l'aplicació es divideix en dos components clarament diferenciats: un component client, que encapsula la interfície modelada, i un component servidor, que encapsula la funcionalitat i l'accés a la Base de Dades. Per tant, una aplicació *OlivaNOVA* està formada per dos components basats en el mateix model conceptual, però independents des del punt de vista de la implementació. Amb l'aproximació OOWS es pretén substituir el component client per la interfície Web implementada amb el framework WIF.

Per acotar el problema i demostrar la viabilitat, la implementació actual del framework només té en compte la plataforma .NET que genera *OlivaNOVA*, on la lògica de negoci s'encapsula com un component COM+, i la base de dades es genera amb SQL Server.

Es pot veure que la integració entre els seus components client i servidor no és diferent que la integració que és pretén en aquest treball. Per tant, pareix raonable aplicar el mateix mecanisme. En les aplicacions generades, aquesta integració es produeix mitjançant una API que exposa la funcionalitat com si les classes foren part del

client. Per cada classe del model, es genera una classe *Proxy* [48] a través de la qual la interfície accedeix a la lògica i la base de dades. Aquesta classe Proxy implementa la lògica necessària per realitzar les crides al component servidor. D'aquesta manera, la lògica de negoci és transparent a la interfície.

Malgrat aço, l'ús d'aquesta API no és possible, i més quan depén de la tecnologia d'implementació. És a dir, si el component servidor és generat mitjançant .NET, les classes de la API estan implementades en .NET, i per tant la interfície ha de ser capaç de treballar amb aquesta tecnologia.

Per tant, s'optà per una altra comunicació que proporciona *OlivaNOVA*: intercanvi de missatges XML de petició/resposta, que consta dels següents passos:

- En primer lloc, el component client construeix un missatge XML mitjançant l'API, per realitzar una operació (de consulta, per exemple).
- Aquest missatge s'envia al component servidor, on es processa i s'invoca la operació sol·licitada.
- El resultat de l'operació es torna a codificar com XML i és enviada al component client com resposta.
- La resposta és rebuda pel component client, que s'encarrega de processar la informació i enviar-la a les classes de la interfície com objectes.

Per tant, podem deduir que el rol de l'API és facilitar la construcció de missatges XML des del punt de vista de la interfície. Des del punt de vista de la generació de codi, aquesta aproximació, igual que el framework, permet abstraure codi complex mitjançant un conjunt senzill de mètodes que realitzen aquesta funció.

La solució adoptada ha sigut la de crear els missatges en la façana de negoci. Quan una classe de la interfície sol·licita una funcionalitat de la capa de lògica, en la façana de negoci es construeix el missatge corresponent i s'envia al component servidor. Una vegada rebut el missatge, aquest és processat en la mateixa façana i emmagatzemat en les estructures de dades que esperen la resta de classes del framework. El XML en el que es basen els missatges és propietari d'*OlivaNOVA*, pero fàcilment comprensible i manipulable.

8.6 Conclusions

Aquest capítol tracta de definir l'estratègia seguida per aconseguir la implementació de la *capa d'Interfície Web* a partir de les extensions conceptuals definides per OOWS, i com aquestes s'integren amb la lògica funcional de l'aplicació.

Primer s'introdueix un estudi conceptual sobre les pàgines web, per tractar de crear una taxonomia amb el tipus de pàgines web més habituals (atenent a la seua orientació) i els tipus de continguts que solen tenir. D'aquesta manera, es pot establir uns vincles entre els tipus de pàgines i els tipus de continguts que haurien de tenir, seguint criteris de qualitat d'aplicacions.

Emprant aquesta informació, es defineix un marc per implementar cada pàgina web, independentment del tipus de pàginas i els tipus de contingut que hi tinga definits. En aquest punt es té en compte, per exemple, la comunicació entre pàgines, la gestió dels usuaris conectats, i el seguiment del contexte o entorn de navegació (la sessió, el camí navegacional, etc.).

També es tracta el tema de la introducció d'aspectes de disseny gràfic a l'hora d'implementar les interfícies d'usuari. A causa de la importància d'aquestos requeriments en les aplicacions web, es defineix una estratègia

en la que puguen descriure's requeriments concrets de visualització, però sense "embrutar" els models conceptuals amb aquest tipus d'informació. Aquesta proposta defineix un marc de treball multi-disciplinar (almenys bi-disciplinar) en el que tant l'Enginyer de Sistemes com el Dissenyador Gràfic (els dos rols principals en aquest punt) puguen dur a terme la seua feina d'una manera independent una de l'altra, però establint uns "contractes". Aquestos contractes es representen mitjançant la definició d'un llenguatge de marcat sobre el que els dissenyadors podran definir les regles de visualització, i que s'obté de constructors conceptuals.

Finalment es presenta l'estratègia de generació de codi aplicant els principis del Desenvolupament Dirigit per Models, en el que els conceptes o primitives dels models són transformats en conceptes d'implementació. Per facilitar la definició d'aquestes regles de transformació, s'ha construït un framework d'implementació d'interfícies Web que captura tots els requeriments abans esmentats (conceptualització de pàgines, estratègia d'implementació de pàgines i introducció d'aspectes de disseny gràfic). Per últim, es descriu com es realitza la integració de les interfícies Web generades amb l'aplicació, per mig de la lògica de negoci.

Capítol 9

OOWS Suite: Eina de Suport

El Desenvolupament de Programari Dirigit per Models (*Model-Driven Software Development (MDSD)*) és una disciplina que està començant a produir resultats interesants. Hi ha diferents indicadors optimístics relacionats amb la implementació industrial d'aquesta filosofia de desenvolupament. Per exemple, la creixent popularitat de l'estàndard MDA [86] proposada pel grup OMG o la filosofia de les Factories de Programari [61] proposades per Microsoft.

Potser, l'indicador més prometedor de resultats és el continu desenvolupament d'eines i tecnologies per construir eines CASE per suportar el MDSD, com ara el Projecte de Modelatge Eclipse [41] i les Eines DSL [78] que estan integrades en el propi Microsoft Visual Studio. Altres eines comercials que donen suport explícit per a MDSD són OptimalJ [33] o AndroMDA [9], que també plantegen un escenari prometedor.

Aquest capítol revisa els aspectes tecnològics que s'han tingut en compte a l'hora de construir l'eina *OOWS Suite* de suport al mètode OOWS. Primer, en la Secció 9.1 es revisen els principis per desenvolupar un entorn que seguisca l'aproximació MDA. En la Secció 9.2, s'analitza l'estat dels Entorns de Producció Dirigits per Models i els seus objectius principal. Amb aquestes premises com base, es presenta l'Editor de

Models (Secció 9.3), que permet fer un tractament de la persistència i manipulació dels models, així com de la visualització d'aquestos. Finalment en la Secció 9.4, s'introdueixen els aspectes considerats per la construcció d'un Compilador de Models i l'estratègia seguida per a aconseguir-ho.

9.1 Definició d'un Entorn de Desenvolupament MDA

Les metodologies de desenvolupament de programari s'estan encaminant cap a l'ús de models no com mera documentació del procés, sinó com elements claus a l'hora d'obenir una implementació definita. A pesa de compartir un estàndar comú, ha aparegut MDA [86] per impulsar el desenvolupament de programari a partir de models, sentant unes nocions bàsiques amplament acceptades.

En essència, MDA és una nova metodologia per la definició d'especificacions i desenvolupament d'aplicacions, que es basa en la transformació de diferents models que van refinant la funcionalitat de l'aplicació fins obtenir el codi final. És precís distingir MDA de MDD (de l'anglés "*Model Driven Development*", "Desenvolupament Dirigit per Models"), que és una aproximació de desenvolupament de programari basat en models conceptuals i la generació de l'aplicació a partir d'aquestos.

Al ser MDA una aproximació, sols recomana una estratègia general a seguir, però sense indicar plataformes, estàndars o processos que ajuden a complir-la. MDA es fonament en tres idees principals:

1. **Representació Directa**: es deu tractar d'apartar el desenvolupament de programari del domini de la solució o representació tecnològica d'aquest, per centrar-se en les idees i els conceptes del domini del problema. Reduint la distància semàntica entre el domini del problema i la seua representació, permet obtenir solucions més precises als problemes plantejats.

2. **Automatització**: emprar eines per mecanitzar totes aquelles tasques del desenvolupament de programari que no necessiten de la creativitat humana. Per això, es deu emfatitzar la transformació entre models i la generació automàtica de codi a partir d'aquestos.

3. **Estàndars Oberts**: el fet que els estàndars siguen oberts facilita que siguen adoptats per la indústria, possibilite la interoperabilitat i que l'usuari final tinga possibilitats d'elecció.

En l'aproximació MDA, en primer lloc es modela un *PIM* ("*Platform-Inependent Model*", Model Independent de Plataforma), que deu capturar la funcionalitat de l'aplicació web de manera independent a la plataforma tecnològica en que serà desenvolupada. Aquest PIM haurà de ser transformat definint les regles corresponents, en un *PSM* ("*Platform-Specific Model*", Model Dependent de Plataforma) que estarà basat en les característiques concretes d'una plataforma d'implementació. A partir d'aquest PSM serà on s'aplicaran les transformacions definitives per aconseguir el codi de l'aplicació.

Per tanto, una aplicació MDA consisteix en un PIM i un o varios PSMs, un per cada plataforma tecnològica destí. La Figura 9.1 a la pàgina següent mostra un resum del procés definit per MDA:

Tant OO-Method com OOWS han sigut dissenyats tenint en compte la seua implementació dins un procés MDA. La implementació comercial MDA d'OO-Method, *OlivaNOVA*, compleix aquestes premises definint un PIM i varios PSM (en forma de motors de transformació) per a diverses plataformes.

L'elecció de MDA com aproximació de desenvolupament ha sigut molt fructífera per l'eina *OlivaNOVA*, demostrant que és possible desenvolupar programari més ràpid i amb menor número d'errors que seguint una aproximació tradicional. Veient l'acceptació industrial d'aquesta eina, es considerà adequat no partir de zero i reutilitzar les idees per definir l'entorn de producció d'OOWS.

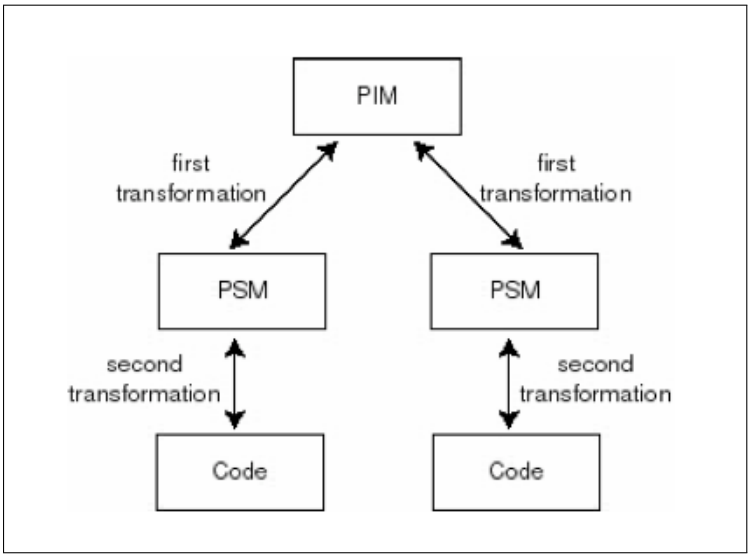


Figura 9.1 – Procés MDA

Seguint aquestes premises, s'ha construit un entorn paral·lel a *OlivaNOVA* que permet treballar amb els Models definits per OOWS, així com integrar les aplicacions generades per aquesta eina amb les solucions generades pels motors de transformació d'*OlivaNOVA*.

Aquest entorn, anomenat *OOWS Suite*, introdueix una sèrie d'eines per a, basant-se en l'aproximació OOWS, fer una extensió conservativa d'OO-Method, per donar suport al desenvolupament d'aplicacions Web. Aquest procés de desenvolupament ha sigut presentat al Capítol 3.

Des d'un punt de vista MDA, el procés que defineix l'entorn pot dividir-se en tres etapes:

1. **Modelatge**: es construeix el PIM que defineix els diferents aspectes d'una aplicació Web. D'una banda, els models OO-Method que capturen els aspectes estàtics i de comportament són definits mitjançant l'editor de models *OlivaNOVA Modeller* [24]. D'altra banda, les extensions web són definides amb ajuda de l'eina *OOWS*

Visual Editor. La Secció 9.3 presenta aquest editor i la interacció amb l'eina *OlivaNOVA Modeller*.

2. **Generació de codi**: en aquesta etapa, es duen a terme dos processos de generació de codi paral·lels. D'una banda, el motor de transformacions d'*OlivaNOVA* (els *OlivaNOVA Transformation Engines*) obté el codi basat en una arquitectura de tres capes a partir de models OO-Method. Per això realitza transformacions PIM-PSM-Codi implementada mitjançant el motor de transformació corresponent [25]. D'altra banda, el *OOWS Model Compiler* genera una interfície Web mitjançant una transformació PIM-PSM a codi. El PSM es troba implementat implícitament al framework WIF (Annexe A). Les transformacions es realitzen agafant el PIM com origen i obtenint codi destí amb el framework.

3. **Integració**: la integració de la informació capturada pels models OO-Method amb la informació capturada pels models OOWS es duu a terme a nivell d'implementació. La interfície Web generada a partir dels models OOWS està basada en el framework, que s'encarrega d'interactuar amb la lògica de negoci d'*OlivaNOVA*

Del resultat d'aquest procés s'obté la lògica de negoci i la persistència (*OlivaNOVA*), explotades per una interfície Web obtesa a partir dels models OOWS emprant el framework WIF.

Per desenvolupar el procés ací descrit, cal definir un entorn que permeta treballar en totes les fases. En el marc del Desenvolupament Dirigit per Models estan sorgint cada vegada més eines que donen suport a aquestos desenvolupaments. En la següent secció s'analitzen les alternatives disponibles i perquè acabem emprant la paltaforma Eclipse.

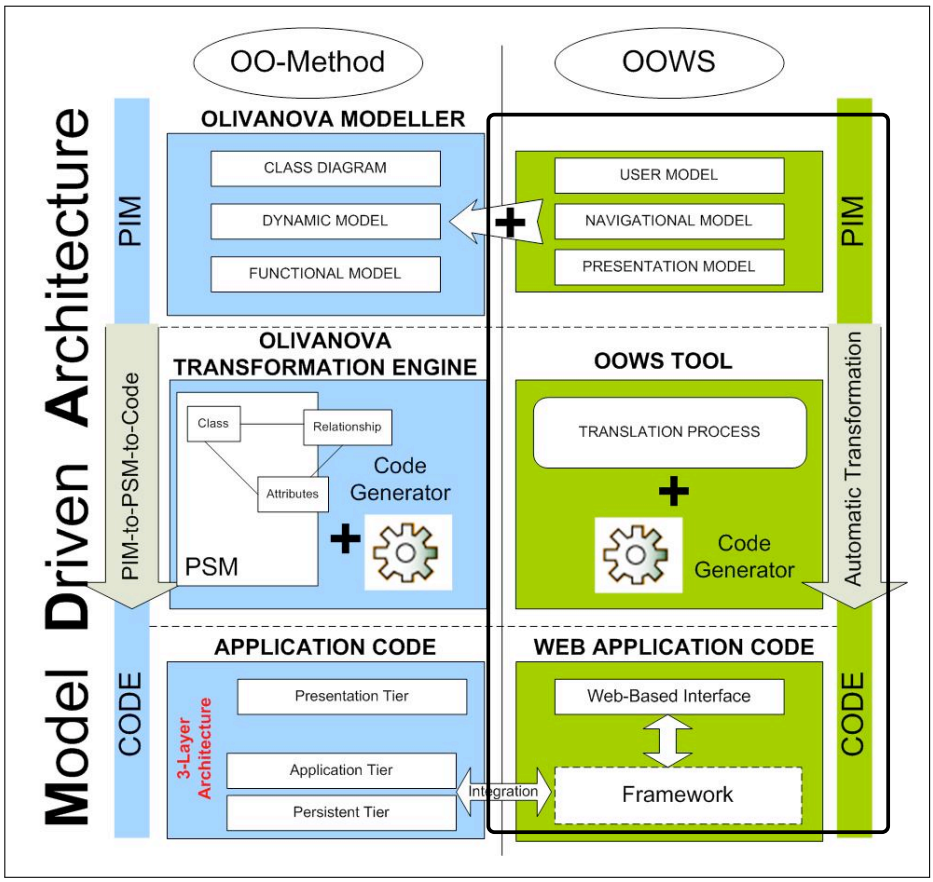


Figura 9.2 – Procés de Desenvolupament MDA de OOWS Suite

9.2 Entorns de Producció Dirigits per Models

Tot procés de desenvolupament ha d'estar suportat per un entorn, és a dir, un conjunt d'eines, editors, etc. que ajuden al desenvolupador a seguir un mètode. En l'àmbit del desenvolupament dirigit per models, tradicionalment s'havia apostat per la personalització d'eines CASE com Rational Rose, ArgoUML o Together. Aquestes eines proporcionen

editors gràfics, generalment basat en UML, que poden ser personalitzats mitjançant estereotipus.

Tanmateix, les funcionalitats de personalització són molt limitades, tant a nivell gràfic, com de generació de codi (on pràcticament són nules). Per aquest motiu, aquestes eines han quedat relegades pràcticament a l'àmbit de la documentació gràfica.

Per fer front a aquesta situació, han començat a sorgir eines per la construcció d'entorns de producció de programari dirigit per models. Aquestes eines no es basen en la personalització d'eines genèriques preexistents, sinó en la construcció de *Llenguatges de Domini Específic* ("*Domain Specific Languages*", DSL) que representen amb fidelitat els conceptes amb que es pretèn treballar.

A partir d'aquestos DSLs, es poden obtenir *editors* (inclús gràfics) per als models, a més d'introduir aspectes de *validació* i *generació de codi*.

Actualment hi ha dos tendències que es troben enfrontades, encara que comparteixen les premises del desenvolupament dirigit per models: l'aproximació proposada pel Projecte Eclipse [41] i les Factories de Programari [61]. D'una banda ens trobem amb un grup d'empreses que recolzen la proposta MDA, entre les que destaca IBM. L'estratègia d'IBM està basada en l'IDE de codi lliure Eclipse, un entorn de desenvolupament de programari extensible que esta sent molt ben acollit per la comunitat de desenvolupadors.

El desenvolupment dirigit per models que proposa IBM és coherent amb les propostes d'estandarització de l'OMG i per tant, segueix els estàndars definits per aquesta. La proposta d'IBM es basa en emfatitzar l'ús de l'UML com estàndar de modelatge, emprant els *Perfils UML* per centrar-se en dominis concrets. Així, a més d'emprar UML com llenguatge de modelatge gràfic, empra XMI (XML Model Interchange) per serialitzar els models i així estandaritzar les eines, i per representar

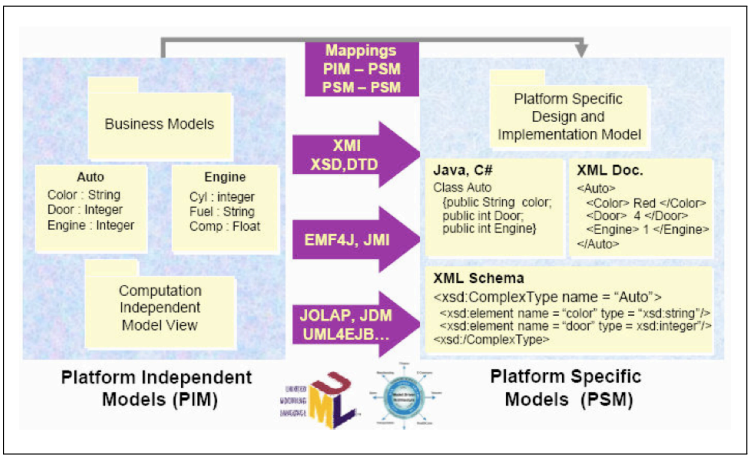


Figura 9.3 – Visió de MDA segons IBM

i manipular els llenguatges de domini específics, aposta per l'ús de MOF o els Perfils UML. Per a la definició de representacions d'informació o esquemes de base de dades, proposa CWM (Common Warehouse Model). La Figura 9.3 mostra la visió de MDA segons IBM i com entren en joc aquestos estàndars.

9.2.1 El Projecte Eclipse

Com nexe comú que proporcione cohesió a tots els estàndars, s'ha definit una extensió per eclipse anomenada "*Eclipse Modeling Framework*" (EMF) [40]. EMF és un framework de modelatge i generació de codi per a la construcció d'eines i aplicacions basades en un model de dades estructurat. Aquest model haurà de ser descrit emprant XMI i a partir d'aquest, EMF és capaç de proporcionar un conjunt de classes en Java que el sustenten. Açò permet, entre altres coses, generar un editor senzill, textual, que serveix per a la visualització i edició del nostre model.

Per a la descripció interna dels models, EMF inclou el seu propi metamodel, basat en MOF, que rep el nom de *Ecore*. A partir d'aquest

metamodel, es proporciona serialització en XMI a una API per a la manipulació dels objectes que el composen.

Una altra característica interessant del framework EMF, és que inclou un conjunt de classes reutilitzable, per a la creació d'editors i models basats en Ecore. La creació d'aquest editor no és limita sols a les classes, sinó que també és possible obtenir una interfície gràfica que el suporte.

Però EMF no proporciona tot allò necessari per a la construcció d'una eina final usable i productiva. Els editors textuals manipulen els models en estructures d'arbre, complicant la manipulació i enteniment de models de cert tamany.

Per sol·lucionar açò, s'està desenvolupant un framework gràfic, "*Graphical Modeling Framework*" (GMF) [42], amb l'objectiu de facilitar la definició i construcció d'editors gràfics. La definició d'un editor gràfic amb GMF es resumeix en la Figura 9.4 a la pàgina següent. A partir d'un metamodel expressat en Ecore (i que defineix el DSL), es defineix la notació gràfica que emprarà l'editor (*.gmfgraph*), els elements per a la creació de les primitives de modelatge (*.gmftool*) i la correspondències entre aquestos elements gràfics i el metamodel (*.gmfmap*).

Una vegada realitzada aquesta especificació, GMF genera de manera automàtica un plug-in d'Eclipse que pot emprar-se com editor gràfic del metamodel (DSL) definit com punt de partida. Aquest editor es genera prenent com a base les tecnologies EMF i GEF ("*Graphical Editing Framework*"), podent ser extés de forma manual mitjançant les possibilitats que ambdós frameworks proporcionen.

9.3 Editor de Models d'OOWS

L'objectiu principal de l'Editor Visual d'OOWS és donar suport a la creació i manipulació dels models conceptuals OOWS. Aquest editor ha sigut desenvolupat emprant Eclipse GMF, cobrint els següents requeri-

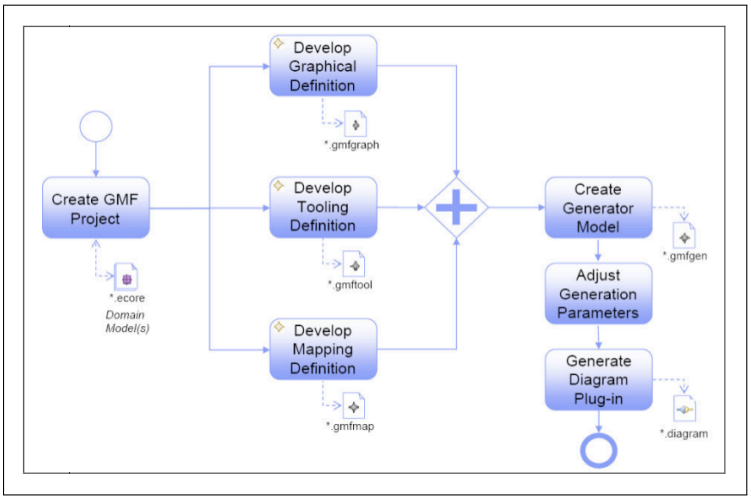


Figura 9.4 – Definició d'un editor gràfic amb GMF

ments fonamentals:

- **Creació i Persistència de Models**: L'editor ha de proporcionar mecanismes que faciliten la creació de models OOWS, respectant sempre el seu metamodel, i la serialització dels models en un llenguatge estàndar.
- **Edició Visual de Models**: Per facilitar la construcció de models era necessari desenvolupar una eina d'edició visual de models, similar a les eines CASE industrials.

9.3.1 Persistència del Metamodel en EMF

Per a la creació i persistència de models, s'han emprat els components que proporciona EMF. En primer lloc fou construida la definició en Ecore a partir del metamodel d'OOWS. Aquest es troba disponible en [128]. A partir d'aquesta definició del metamodel, EMF proporciona automàticament un entorn per poder crear i modificar els models, de

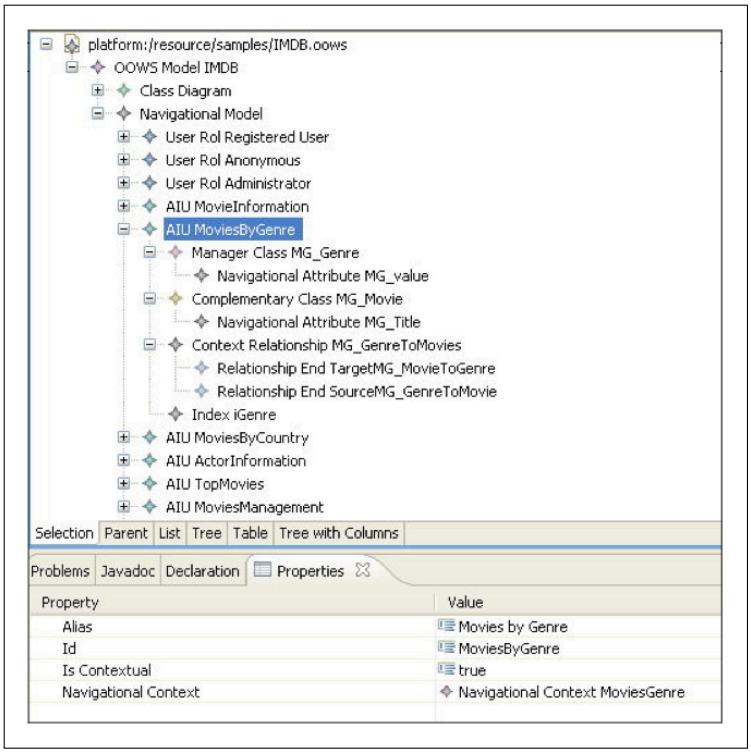


Figura 9.5 – Editor EMF del Metamodel d'OOWS

manera textual en forma d'arbre. Aquest editor disposa d'algunes funcionalitats avançades, com la possibilitat de restringir la cardinalitat o valors possibles de les relacions, validar el model o mostrar diferents vistes del model. A més a més, EMF emmagatzema els models en XMI, obrint el camí a l'intercanvi de models amb altres eines.

La Figura 9.5 mostra l'editor generat automàticament per a una especificació OOWS.

9.3.2 Editor Gràfic en GMF

Per a la construcció de l'editor gràfic, es trià GMF. Aquest editor està actualment subdividit en tres editors, que cobreixen un part concreta del metamodel: l'*editor de Models d'Usuari*, l'*editor de Mapes Navegacionals* i l'*editor de Unitats d'Abstracció d'Informació*.

Ja que la notació gràfica d'OOWS és molt semblant a la d'UML, gran part de la notació gràfica ja era proporcionada d'entrada per GMF, com la definició de les classes contenidores, les fletxes, els paquets, etc. Tanmateix, l'editor no ha sigut completat a causa de restriccions en la plataforma, anuncis de canvis en versions immediates i l'estat encara no suficientment madur de GMF. Aquestes restriccions implicaven, de vegades, la modificació del metamodel (Ecore) per poder representar gràficament alguns constructors. És per açò que l'editor sofrirà millores en un futur proper quan GMF avance en el camí que es planteja.

Tanmateix, tot allò que de moment no pot ser representat gràficament, pot gestionar-se mitjançant l'editor textual que proposa EMF, que és completament funcional.

La definició dels models OOWS es realitza partint del model estructural d'OO-Method definit mitjançant l'*OlivaNOVA Modeller*. Les UAIs que formen els contextes poden definir-se com vistes sobre les classes, atributs i relacions d'aquest model OO-Method. Així doncs, la interacció entre ambdues eines de modelatge es durà a terme a través del model estructural d'OO-Method.

Per començar a crear els models OOWS, és necessari importar el diagrama de classes d'OO-Method a l'entorn de l'eina. La importació directa d'un model creat en *OlivaNOVA Modeller* no és possible en Eclipse ja que el XML en que s'emmagatzema segueix un esquema propietari. Si bé l'eina inclou mecanismes d'exportació a XMI [73], la versió de XMI especificada no és compatible amb Eclipse.

Per tant, és necessari definir un mecanisme per importar aquestos

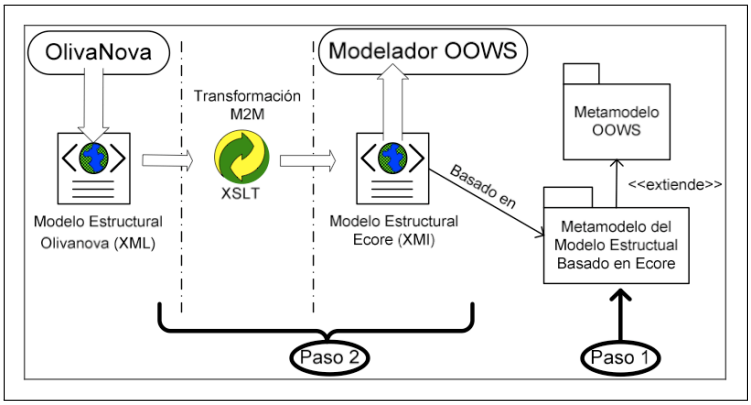


Figura 9.6 – Importació entre *OlivaNOVA* i *OOWS Suite*

models definits amb l'*OlivaNOVA Modeller*. Per complir aquesta tasca, s'han d'implementar dos tipus de mecanismes: (1) mecanismes que faciliten a l'eina *OlivaNOVA* l'exportació del model estructural, i (2) mecanismes que faciliten als editors gràfics d'OOWS la importació del model.

Com que *OlivaNOVA* emmagatzema el seus models en XML, s'ha fet que siga aquest document el que servisca d'entrada als editors. Així, la solució adoptada ha sigut processar aquest XML amb una plantilla de transformació XSLT, de manera que s'obtinga com a resultat el metamodel d'*OlivaNOVA* representat en l'Ecore d'OOWS, obviant aquelles parts del model original innecessàries. La Figura 9.6 mostra esquemàticament aquest procés d'importació.

Encara que el model estructural és importat a l'*OOWS Visual Editor* i que aquest podria modificar-se, cal tenir en compte que la lògica de negoci que genera *OlivaNOVA* depèn d'aquest model, i no s'ha definit cap procediment per que *OlivaNOVA* l'importe. Així, les modificacions a l'estructura d'informació del sistema estan prohibides a l'eina *OOWS Visual Editor*, i hauran de realitzar-se amb l'*OlivaNOVA Modeller*.

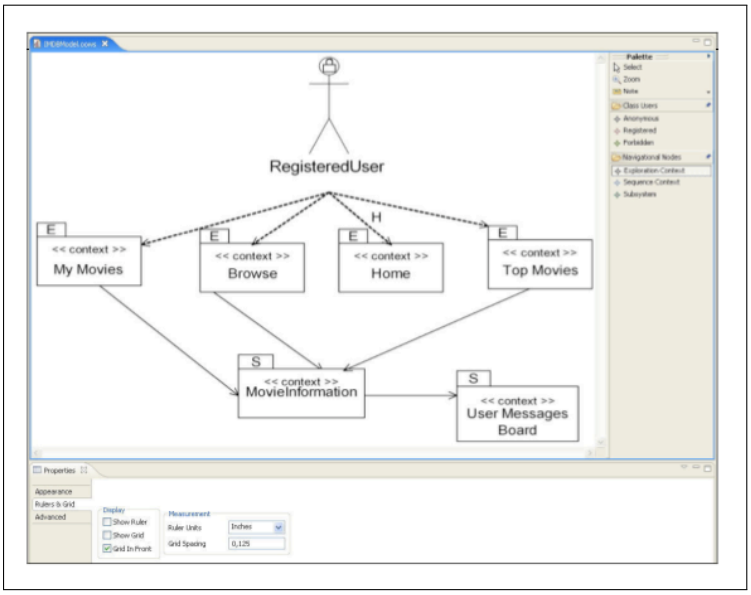


Figura 9.7 – Editor GMF de Mapes Navegacionals

A efectes d'il·lustrar l'eina, es mostra una sèrie de captures de pantalla de l'aspecte de l'editor de mapes navegacionals (Figura 9.7) i l'editor de contextes (Figura 9.8 a la pàgina següent). Tanmateix, aquesta es troba disponible en [128].

9.4 Compilador de Models OOWS

Definim el compilador de Models OOWS com un conjunt de regles de transformació, de PIM a codi, que a partir d'una especificació OOWS, genera una interfície Web i la funcionalitat necessària per integrar-se amb la lògica de negoci implementada per l'eina *OlivaNOVA*. Una vegada definits tots els models, falta decidir com es realitzaran les transformacions entre aquestos models.

La OMG llançà una petició de propostes per definir un nou estàndar,

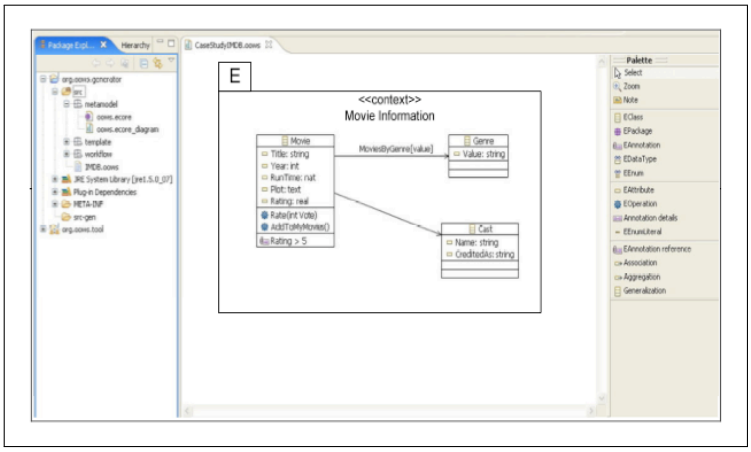


Figura 9.8 – Editor GMF de Contextes Navegacionals

sota el nom de MOF2Text [85], l'objectiu del qual era precisament servir de llenguatge per a la transformació de models definits en UML, a text. Tanmateix, a data d'avuí encara està en fase d'aprovació i no existeix cap implementació d'aquest. Però existeixen alternatives per dur a terme aquesta tasca.

En [112] ens descriu com emprar tècniques de **reescritura de gramàtiques** de grafs per realitzar transformacions de model a codi. Aquesta tècnica es basa en definir una sèrie de regles de reescritura de tal manera que quan un graf compleix una regla, es reemplaçat per un nou graf que el representa en el nou model. Aquesta aproximació té diversos avantatges: d'una banda, les transformacions poden representar-se de manera visual (una regla, una transformació); d'altra banda, té una sòlida base formal. L'eina AGG Tool [3] simplifica el procés de definició de regles i emmagatzema els resultats en XML, facilitant la interoperabilitat. A mode d'example d'ús exitós d'aquestos entorns, en [131] s'empra aquesta eina per definir i realitzar transformacions entre un metamodel de requeriments web i el metamodel d'OOWS. D'aquesta manera, s'a-

consegueix obtenir models navegacionals a partir de requeriments web mitjançant reescritura de grafs.

El principal problema d'aquesta aproximació per a l'ús que necessitem és que aquestes aproximacions serveixen per realitzar transformacions model-a-model (o *PIM-a-PSM*. Tanmateix, com s'ha discutit a la Secció 9.1, les transformacions que pretenem realitzar seran model-a-codi (o *PIM-a-Codi*). És per açò que aquesta aproximació no és la més adient.

Una altra possibilitat era la d'emprar **plantilles XSLT** igual que en [69]. En aquest treball s'empren una sèrie de transformacions XSLT per compilar una sèrie de models UML emmagatzemats com XML. De la mateixa manera que l'XML, el XSLT és un llenguatge proposat per la W3C que és amplament suportat i estandaritzat, podent-se emprar amb nombrosos llenguatges i entorns de desenvolupament. A més a més, és un llenguatge flexible i relativament potent a l'hora de definir transformacions en XML.

El principal inconvenient de realitzar transformacions basades en plantilles XSLT consisteix en que la creació i manteniment de regles és una tasca costosa. A causa de les característiques sintàctiques del XSLT, la plantilla resultant és difícil d'entendre i modificar. Açò és degut a que, a l'igual que l'XML, el seu model subjacent és un model entitat-relació, i no orientat a objectes. Açò provoca que tant la representació de la informació, com l'encapsulació de les regles de transformació no siga massa adients ni fàcilment manipulables. Si el resultat de la transformació no és XML, com és el cas que ens ocupa, el problema encara s'agreuja més.

Una tercera aproximació per a la generació de codi és la utilització de **plantilles** igual que en [79]. Una plantilla és un troç de codi que deu ser completat a partir de la informació que conté un arxiu origen, com per exemple, un model emmagatzemat en XML. Amb aquest objectiu,

els llenguatges de transformació basats en plantilles que proporcionen la informació a la plantilla definint una espècie de meta-programació.

Existeixen nombrosos llenguatges de plantilles, tals com *Velocity* [10], *FreeMarker* [47] o JET, el llenguatge que empra EMF per produir automàticament els editors EMF/GMF. Aquest tipus de llenguatges són més potents en el sentit que es poden definir les transformacions d'una manera molt pròxima al codi a generar. Tanmateix, el seu manteniment pot considerar-se de l'estil del XSLT, costós a causa de barrejar codi final amb codi per definir la plantilla.

L'alternativa a aquestos llenguatges, són els llenguatges de transformació *model a text*, com *MOFScript* [43] o *oAW Xpand* [127]. A diferència dels llenguatges de plantilles anteriors, aquestos llenguatges tenen en compte que l'entrada no és un document qualsevol, sinó un model conceptual. Aquest tipus de llenguatges ofereixen una visió més declarativa, en contrast amb els anteriors tipus de llenguatges de plantilles, que tenen una visió més imperativa. A més, entre les facilitats de transformació, introdueixen operadors que permeten navegar i obtenir elements del model de manera senzilla. Per tant, aquesta és l'opció que més s'ajusta a les necessitats de l'entorn a desenvolupar. A dia d'avuí, oAW ofereix un major número de possibilitats amb respecte a MOFScript.

9.4.1 Transformacions amb OpenArchitectureWare

OpenArchitectureWare (oAW) [127] és un Framework de suport al Desenvolupament de Programari Dirigit per Models i MDA, compost per diferents mòduls que cobreixen gran part del procés de desenvolupament. El Framework suporta diferents tipus de metamodels i proporcioan un conjunt de llenguatges per validar, transformar i generar codi a partir d'aquestos.

oAW manté una estreta relació amb el *Projecte Eclipse*, ja que el

seu IDE s'integra amb la plataforma Eclipse i suporta models definits en EMF de manera nativa, així com models basats en UML o XML.

El nucli de oAW és un motor de "*workflow*" que duu a terme el procés de generació de codi. Diversos components predefinits poden ser incorporats al workflow per a la lectura i instanciació de models, la seua verificació, la transformació entre models i finalment, la producció de codi.

S'ha triat oAW bàsicament per la seua versatilitat. A pesar que només s'ha emprat el component de generació de codi *Xpand*, l'ampli ventall de possibilitats que ofereix oAW permet que en un futur es puga millorar aquest procés de generació.

Per crear un generador de codi amb oAW Xpand, en primer lloc s'ha de definir el fitxer del *workflow* del procés de generació. Aquest workflow controla les diferents fases: càrrega dels models, validació i generació. A més, permet configurar el metamodel en que es basa la generació (en aquest treball s'empra el metamodel d'OOWS en Ecore), així com el directori destí del codi generat. Important en aquest punt és definir la plantilla que s'emprarà com *arrel* ("*Root*") a partir de la que comença el procés de transformació.

Les regles del compilador es troben definides com plantilles escrites en Xpand. El concepte central d'aquest llenguatge és el bloc DEFINE, que representa la unitat mínima que forma una plantilla. Cada bloc DEFINE especifica una regla de transformació d'un element del metamodel. La capçalera d'un DEFINE es composa d'un identificador de la regla, un conjunt de paràmetres (opcionals) i el nom de la classe del metamodel sobre la que es defineix la regla. En certa manera, aquestes regles de transformació poden considerar-se com mètodes de la metaclase sobre la que es defineixen.

Xpand incorpora una propietat this implícita, a partir de la qual es poden referenciar, mitjançant el seu identificador, els atributs de la

metaclase i altres elements del metamodel relacionats. El cós del bloc DEFINE pot contenir altres expresions del llenguatge o qualsevol tipus de text que s'afegirà al procés de generació. Per diferenciar el codi estàtic a generar de les expressions i instruccions de Xpand, aquestes últimes es delimiten mitjançant els símbols ≪≫. Les expresions són les encarregades de processar la informació proporcionada per la instanciació del metamodel.

Xpand proporciona operadors avançats per la selecció, manipulació de coleccions i navegació de la informació del metamodel. Després d'avaluar una expressió, el resultat és bolcat com text.

Una característica interessant de la definició de regles en Xpand és el suport al polimorfisme. Si existeixen dues regles amb el mateix nom definides sobre dos metaclasses que hereten de la mateixa metaclasse pare, Xpand selecciona de manera automàtica la regla adequada, encara que s'haja invocat com si fós la metaclasse pare. Si no existeix una regla que redefineix la metaclasse filla, per defecte s'invoca a la regla que pertany a la metaclasse pare.

La generació de codi emprant Xpand es basa en la concatenació de crides a regles que produeixen codi per a un element concret del metamodel. Aquest procés comença en la regla DEFINE marcada com "*Root*" al fitxer del workflow. Per invocar una regla, cal emprar la sentència EXPAND, que rep el nom del bloc DEFINE a expandir, junt als paràmetres si aquest en tinguera.

La sentència EXPAND també rep un element del metamodel que deu ser del tipus que espera el bloc DEFINE. Aquest element pot referenciar-se mitjançant l'operador FOR si és *únic*, o mitjançant FOREACH si és una col·lecció d'elements (obtesos per exemple mitjançant una relació de cardinalitat n). A aquestos elements obtesos d'aquesta manera se'ls aplicarà la regla.

Molt important en aquestos llenguatges de plantilles és definir el

```
«DEFINE Root FOR OOWSModel»
<?php
$Application = new Application("«id»","ONME","«id»");

«FOREACH NavigationalModel.UserRol AS Rol»
 «IF Rol.id == "Anonymous" »
       $Application->AllowAnonymous();
 «ELSE»
  $Application>Rol("«Rol.id»","«Rol.Class.id»",
  "«Rol.Class.id»_Login","«Rol.Parent.id»");
 «ENDIF»
 «EXPAND NavigationalNodes_rule FOREACH Rol.NavigationalNode»
«ENDFOREACH»
$Application->HomePage("«NavigationalModel.Root.id»");
?>
«ENDDEFINE»

«DEFINE NavigationalNodes_rule FOR NavigationalContext»
$Application->Page("«id»","«alias»","«UserRol.id»","«reachability.Access()»");
       «EXPAND ContextTemplate::Context_Rules FOR this»
«ENDDEFINE»

«DEFINE NavigationalNodes_rule FOR NavigationalSubsystem»
       $Application->PageGroup("«id»","«UserRol.id»","Root");
       «EXPAND NavSubsystem_rule FOR this»
«ENDDEFINE»
```

Figura 9.9 – oAW Xpand: Exemple de Regla de Transformació

fitxer d'eixida on es generarà el codi. Xpand utilitza la sentència FILE per a aquest propòsit. Aquesta sentència s'empra dins les definicions de les regles, i se li passa com a paràmetre el nom del fitxer que es vol obtenir. És habitual que el nom del fitxer es genere a partir de la pròpia informació del model d'entrada a les transformacions.

El llenguatge Xpand també incorpora instruccions pròpies d'un llenguatge imperatiu, com ara bucles foreach o sentències condicionals if, que ajuden a estructurar la definició de les regles.

La Figura 9.9 mostra un exemple de codi que ilustra els conceptes ací detallats. Aquesta regla forma part de la plantilla *arrel* del procés de generació, i està composada per dos regles: *Root* i *NavigationalNodes_rule*.

La regla **Root** inicia el procés de generació rebent com entrada l'element del metamodel *OOWSModel*, que també és l'arrel del meta-

model Ecore d'OOWS. Fora de les sentències del llenguatge Xpand, pot observar-se el text que anirà produint-se en la plantilla. Aquest text correspon a codi en *PHP* que empra el *Framework d'Interfícies Web* descrit en l'Annexe A. Aquest codi es genera de maera estàtica, com les capçaleres del fitxer PHP o les crides a l'objecte *$Application* del Framework.

L'objectiu d'aquesta regla *Root* és recòrrer els rols (usuaris) que formen part del Model Navegacional (*NavigationalModel.UserRol*, segons el metamodel d'OOWS en Ecore) de l'aplicació, mitjançant el bucle FOREACH. Si detecta el Rol *Anonymous* (comprovat mitjançant la instrucció IF), a més genera la crida al Framework per habilitar la autenticació anònima (*$Application->AllowAnonymous()*).

A continuació, i per a cada Node Navegacional que pertany a un Rol (*Rol.NavigationalNode*), s'invoca a la regla *NavigationalNodes_rule* mitjançant la sentència EXPAND. Aquesta regla s'encarrega de produir el codi associat per a cada node. Com es pot apreciar, en aquesta regla s'aplica el polimorfisme d'oAW que s'ha comentat abans, per diferenciar la generació de codi entre un *Navigational Context* i un *Navigational Subsystem*, ambdues subclasses del *Navigational Node* al metamodel d'OOWS.

La regla *NavigationalNode_rule*, aplicada a un *NavigationalContext* crea una nova pàgina web mitjançant la instrucció el framework *$Application->Page(...);*.

La regla *NavigationalNode_rule, aplicada a un NavigationalSubsystem crea un grup de pàgines ($Application->PageGroup(...)) i després invoca a la regla que processa el Subsistema (NavSubsystem_rule).*

Normalment els metamodels són descrits d'una manera estructurada, ja siga de manera gràfica o a través d'una jerarquia (com és el cas de l'"arbre" amb que ho representa EMF). A causa d'aquesta restricció, de vegades es complexe explorar el model i definir comportament addicional,

com ara operacions de consulta o atributs derivats a partir d'altres. A més, és una bona pràctica no incloure en el metamodel informació relativa al procés de transformació.

Per donar una solució a aquestos problemes, Xpand introdueix una sèrie d'extensions basades en el llenguatge *Extend*. Una extensió és una propietat complexa definida sobre un element del metamodel, de manera externa a aquest. A través de les extensions és possible afegir comportament addicional que puga ser reutilitzat i que simplifica encara més la definició de les regles.

El següent exemple mostra el codi de l'extensió *Access* emprada en la regla*NavigationalNode_rule* sobre l'element del metamodel *Reachabilities*. En funció del valor d'aquesta propietat, retorna la cadena "Always" o "FromPage", que són els tipus d'accessibilitat que el Framework empra:

```
String Access(Reachabilities r):  r == Reachabilities::Exploration ?
"Always":  "FromPage";
```

Aquest procés s'ha aplicat per a cadascun dels elements del metamodel d'OOWS de manera que per a cada primitiva conceptual (elements del metamodel) s'ha definit una regla DEFINE. La semàntica de les transformacions pot veure's al Capítol 8 a la Secció 8.5. L'Annexe D mostra el codi complet del compilador de models OOWS (a patir de l'Ecore del metamodel d'OOWS) i la seua transformació a codi del Framework (Annexe A), emprant oAW Xpand.

Capítol 10

Conclusions

Aquesta tesi ha presentat *OOWS*, un *Mètode per al Desenvolupament d'Aplicacions Web dirigit per Models*, que permet realitzar extensions conceptuals introduint els *conceptes Web* bàsics i defineix una estratègia aplicant *compilació de models* per obtenir aplicacions funcionals de manera automàtica. A més a més, es caracteritza perque per dur-ho a terme, s'han unit els principis de l'*Enginyeria Web* i materialitzats mitjançant l'aplicació de les tècniques de *Desenvolupament de Programari Dirigides per Models i MDA*.

La xarxa *MDWE.NET* [136, 66] ha sorgit recentment de la mà dels creadors dels principals mètodes de desenvolupament Web on defenen i aposten clarament per aquesta estratègia. Açò reforça encara més la línia d'actuació presa en aquest treball, on des d'un principi s'ha apostat per l'ús i aplicació d'aquest tipus d'entorns.

En aquest últim capítol s'introdueixen les conclusions del treball realitzat en aquesta tesi. En primer lloc es revisen les principals contribucions en l'àrea de l'Enginyeria Web. A continuació es presenta l'investigació actual i treball futur directament relacionat. Per últim, les contribucions en aquesta àrea en forma de publicacions d'investigació.

10.1 Principals Contribucions

Les aportacions principals d'aquesta tesi es poden resumir breument com:

- En primer lloc, la definició un **procés complet** per construir aplicacions Web. En aquest procés es consideren tant les etapes primeres per a la construcció de Models Conceptuals, com l'etapa de Implementació, on es construeix de manera sistemàtica una aplicació funcional a partir d'aquestos models. Aquest s'ha definit tenint en compte els *principis* que proposa l'**Enginyeria Web** on s'emfatitza la necessitat de definir processos sistemàtics on s'introdueixen els conceptes Web als mètodes convencionals per **desenvolupar sistemàticament** les aplicacions Web.

- A nivell conceptual, s'han revisat els requeriments navegacionals de les aplicacions i s'ha proposat una **extensió a OO-Method**, introduint-hi diferents models (d'usuaris, navegació i presentació) per poder obtenir **Models Conceptuals Web**.

- Gràcies al fet que s'ha emprat OO-Method com a mètode subjacent i que aquesta té una forta **base formal**, la representació dels requeriments navegacionals, ha pogut dur-se a terme d'una manera més precisa que aquells mètodes que empren UML com a base. Així, s'han establit unes fortes relacions entre les primitives d'ambdós mètodes, aconseguint una **solució cohesionada**. Així, s'han pogut resoldre problemes que encara avuí en dia tenen una representació complicada en altres aproximacions, com ara el tractament d'aspectes dinàmics-funcionals de les aplicacions (a causa principalment de l'UML i que ha afectat per extensió la resta de mètodes) que en aquesta aproximació OO-Method/OOWS no ha sigut problemàtica.

- S'ha realitzat un estudi intens sobre les primitives que represent els requeriments de navegació, analitzant les amplament acceptades. Sobre aquestes s'han introduït algunes millores, que si bé subtils, introdueixen beneficis. Així, per exemple, la revisió dels **mecanismes d'accés** (índexs, filtres) establint millors criteris a l'hora de definir-los i convertir-los en accessoris per millorar la navegabilitat, enlloc de definir-la. També la introducció del concepte d'**herència navegacional**, que permet d'una banda reutilitzar especificacions conceptuals i d'altra simplificar la tasca de definició dels models navegacionals.

- Per aconseguir un millor enteniment dels models proposats i per permetre la interacció entre aproximacions (molt en vigor actualment), s'ha definit el **metamodel** d'OOWS. Aquest metamodel recull el significat de les primitives i les inter-relacions entre aquestes, tant a nivell navegacional, com a nivell d'estructura d'informació. El metamodel d'OOWS està servint també com a la base per a la seua extensió, facilitant la introducció dels conceptes que es revisen a la Secció 10.2.

- Com que en el desenvolupament d'aplicacions Web és molt important l'aspecte visual del producte resultat, en aquest treball s'ha realitzat una proposta d'**integració d'aspectes de disseny visual** que permeten realitzar un treball multidisciplinar en conjunt amb experts en el disseny gràfic. Però, aquesta aproximació permet que la tasca realitzada per ambdues disciplines puguen dur-se a terme de manera ortogonal i separada. Un dels punts claus per aconseguir-ho ha sigut la definició d'aquestos requeriments fora dels models de l'aplicació (a diferència d'altres) però establint clarament el "contracte" a seguir per a la seua definció.

- Per facilitar la implementació de les aplicacions Web, s'ha realitzat una anàlisi de les pàgines Web que composen aquestes aplicacions, atenent als seus *objectius* i *continguts*. Açò ha permés donar una **conceptualització de les pàgines Web**. Atenent a aquesta conceptualització, i seguint els principis de les *Factories de Programari*, s'ha desenvolupat un **Framework d'Interfícies Web** que recull aquestes idees i proporciona un mecanisme per implementar, d'una manera senzilla, les aplicacions Web.

- Una vegada conceptualitzades les pàgines Web, s'han seguit els principis de la transformació de models descrits en *MDA* per definir una **estratègia de compilació de models a codi**. S'han definit les correspondències que permeten realitzar la **compilació de models OOWS**, realitzant les transformacions de les primitives d'una manera *abstracta*, de manera que puga ser aplicada a la generació en qualsevol plataforma.

- S'ha construït un entorn, *OOWS Suite*, que materialitza aquestes idees. Per a la seua construcció s'han aplicat les últimes tendències tecnològiques aplicades al *Desenvolupament Dirigit per Models*, i emprant l'entorn proposat pel *Projecte Eclipse*, s'han definit: un **gestor de repositoris de models** amb EMF (encarregat de mantenir i manipular els models), un **editor gràfic** amb GMF (que permet capturar mitjançant la seua representació gràfica, els diferents models Web) i un **compilador de models** en oAW Xpand que aplica les transformacions Model a Codi de manera automatitzada.

- S'ha provat la valia d'aquest mètode en el desenvolupament de diferents **aplicacions reals**. Altres vegades, el mètode s'ha emprat per fer enginyeria inversa i poder documentar, entendre, millorar el desenvolupament o estimar el cost del desenvolupament d'una

aplicació. Aquestos són alguns del desenvolupaments realitzats:

- *Web DSIC*: una part de l'aplicació Web d'aquest departament fou desenvolupada amb aquest mètode [132].
- *BibioUP*: es construí una intranet que gestiona els recursos bibliogràfics i personal en la UPV
- *Centre a la Cooperació i Desenrollament (UPV)*: aquesta aplicació Web, de caràcter solidari, ajudà en la definició i introducció dels aspectes de disseny gràfic.
- *Web del Grup OO-Method*: aquesta aplicació permet la gestió d'un grup d'Investigació (membres, publicacions, projectes, activitats, recursos, etc.)
- *Peritatge de la Web del València C.F.*: es realitzà un peritatge de la web del València C.F. per estimar el cost de desenvolupament. Per a realitzar-ho, es construiren els models conceptuals OO-Method/OOWS i s'estimà el seu cost aplicant una proposta d'estimació del tamany funcional a partir de models OOWS [1].

Quasi totes aquestes contribucions han sigut establertes mitjançant la publicació de diferents articles en conferències de la temàtica adient (en cada cas) per donar-li la validesa científica necessària. A la Secció 10.3 queden recollides aquestes publicacions.

10.2 Investigació Actual i Treballs en Curs

Durant els últims anys, el mètode OOWS ha anat evolucionant incloent característiques per donar suport a noves extensions. Aquestes extensions tenen a veure amb els següents tòpics: *Modelatge de Requeriments Web*, *Modelatge de Sistemes Adaptatius*, *Modelatge de Processos de Ne-*

goci, *Aplicacions de les tecnologies de la Web Semàntica* i *Arquitectures Orientades a Serveis*.

10.2.1 Modelatge de Requeriments Web

Aquesta extensió proposa l'ús d'un model de requeriments per al desenvolupament d'aplicacions Web [131]. Aquest model es basa en el concepte de tasca, reorientant-les per capturar no sols aspectes estructurals i de comportament (com esdevé en aplicacions no-Web), sinó també per tenir en compte requeriments navegacionals.

Aquest model de requeriments introduït a OOWS es crea en 3 grans etapes [134]:

1. En una primera etapa es crea una *taxonomia de tasques*. Aquesta taxonomia especifica una estructura jeràrquica de tasques que els usuaris han de acomplir quan interactuen amb l'aplicació Web. Hi ha tasques *generals* i tasques *específiques*. Per tal de dur a terme un refinament de tasques, es proposa l'aplicació de descomposicions estructurals i temporals.

2. En una segona etapa cada tasca bàsica és descrita analitzant l'interacció amb l'usuari. S'empra una estratègia basada en *Diagrames d'Activitats*. Cada diagrama d'activitats es definit emprant accions del sistema i punts d'interacció, que representen el moments durant una tasca on el sistema i l'usuari intercanvien informació (aquesta informació ens permet capturar la semàntica navegacional a nivell de requeriments).

3. En una tercera etapa es descriuen una sèrie de *plantilles d'informació*. Aquestes plantilles descriuen la informació que s'intercanvia en cada punt d'interacció.

Després, s'aplica una estratègia per realitzar transformacions Model-a-Model per obtenir models navegacionals parcials a partir dels requeriments definits. El procés de generació de codi permet obtenir prototipus en entorns multidisciplinars definint diferents rols (dissenyadors gràfics, experts en usabilitat, etc.) i responsabilitats.

Les transformacions Model-a-Model i Model-a-Codi definides proporcionen un mecanisme d'alt nivell per traçar des dels requeriments fins al codi. Aquesta característica facilita la gestió dels requeriments volàtils i l'evolució de l'aplicació.

10.2.2 Modelatge de Sistemes Adaptatius

La major part dels esforços en l'àrea de la hipermèdia adaptativa es centren en implementar conceptes d'adaptativitat per resoldre problemes i en desenvolupar i millorar les estratègies d'adaptativitat i algorismes, que són introduïts en fases posteriors del procés de desenvolupament de programari.

En aquest treball es proposa la inclusió de perspectives de major nivell d'abstracció (més generals i independents del domini) per al desenvolupament d'aplicacions de la hipermèdia adaptativa, dins del marc del desenvolupament dirigit per models. Tanmateix, encara hi ha problemes importants amb respecte al poc suport conceptual per a múltiples tècniques d'adaptativitat, i la carença completa d'un suport metodològic.

L'aproximació que es proposa per dotar a OOWS de suport per al desenvolupament d'aplicacions web adaptatives [101] introdueix eines conceptuals per descriure aquestes tècniques de l'adaptativitat, a un nivell d'abstracció elevat. Per tal de donar aquest suport, les principals parts de l'extensió d'OOWS han sigut:

1. Una estratègia de modelatge d'usuaris basada en *la descripció d'aquestos usuaris* com conceptes propis del domini, considerant

les seues característiques personals, les seues relacions amb el domini de l'aplicació, i la descripció de la seua interacció amb l'aplicació [104].

2. Un conjunt d'estructures i propietats conceptuals, incorporades al model navegacional d'OOWS, que donen suport a tècniques adaptatives ben conegudes, com ara l'ocultació d'enllaços ("*link-hidding*"), l'ordenat d'enllaços ("*link-ordering*"), o fragments condicionals [103].

3. Una aproximació d'especificació de requeriments que inclou capacitats per definir els requeriments relatius als diferents usuaris de l'aplicació (requeriments de classificació d'usuaris, informació i funcionalitat). A més a més, proporciona eines per descriure el conjunt dels requeriments d'adaptativitat del sistema, en termes de característiques de tasques del sistema (vist a la Secció 10.2.1). En aquest sentit, les decisions sobre l'adaptativitat es prenen en la fase de modelatge conceptual i són suportades pels requeriments d'adaptativitat a usuaris [104].

4. Una aproximació sistemàtica per derivar especificacions conceptuals de característiques d'adaptativitat a partir de les especificacions de requeriments corresponents, per mig de la definició de regles de mapeig entre les estructures dels models conceptuals d'OOWS [104].

10.2.3 Modelatge de Processos de Negoci

L'increment en l'ús de la tecnologia dels Serveis Web ha fet d'Internet una plataforma molt adient pel desenvolupament d'aplicacions de negoci (moltes companyies ja proporcionen serveis a terceres companyies emprant aquestes tecnologies).

Un dels problemes principals sorgeixen quan aquest tipus d'aplicacions són com les següents: (1) de vegades, la descripció d'aquestes aplicaciones de negoci està altament lligada a *descripcions de processos de negoci*, on l'objectiu d'aquestes aplicacions no és sols la gestió d'informació sinó també la gestió del propi procés; (2) els processos de negoci reals no sols inclouen tasques automatitzades i el propi sistema; de fet, aquestos processos poden incloure *participants humans* (participants que requeriexen una interfície d'usuari per interactuar amb el procés) i *activitats manuals* (activitats que no són automatitzables en absolut, com ara, fer una crida telefònica o revisar un document).

Des de la perspectiva que proposa OOWS, s'està realitzant una extensió per generar aplicacions Web que donen suport complet a l'execució de processos de negoci [129]. Per tal d'aconseguir-ho, es proposa generar a partir d'especificacions de processos de negoci: (1) la interfície gràfica d'usuari per llançar i completar les activitats del procés, així com (2) la definició d'un procés executable equivalent.

Tanmateix, aquesta aproximació força a revisar l'aproximació OOWS no sols des d'un punt de vista de modelatge, sinó també des d'un punt de vista arquitectònic (en alguns casos, la execució d'un procés es realitza en un motor de processos de negoci).

Aquesta aproximació permet obtenir implementacions de processos de negoci que estan totalment integrats dins aplicacions Web. Aquesta integració es produeix a tres nivells: *dades/contingut*, *funcionalitat* i *interfície d'usuari*. Per aquest motiu, s'ha definit una extensió sobre el model navegacional per modelar aquestes interfícies d'usuari que són necessàries per descriure la interacció entre els participants humans i el procés de negoci.

10.2.4 Aplicacions de les tecnologies de la Web Semàntica

Amb l'objectiu de dur a la realitat la Web Semàntica [14], és necessari proporcionar guies, mètodes i eines que remarquen l'ús i aplicabilitat de les tecnologies de la Web Semàntica en desenvolupament d'aplicacions reals. El desenvolupament en aquest sentit no sols involucra la generació de contingut semàntic (definint dominis específics mitjançant ontologies), sinó que també anotar la semàntica de funcionalitat que permeta a usuaris externs i aplicacions de programari, *descobrir, invocar, composar i monitoritar* aquesta funcionalitat amb un alt nivell d'automatització.

En aquest sentit, els mètodes d'Enginyeria Web deuen estar preparats per proporcionar solucions que consideren el modelatge d'aquesta dimensió, referint-se a la vista del sistema des d'un punt de vista de la Web Semàntica. Aquesta nova dimensió permet generar aplicacions no sols per ser emprades per humans, sinó també que puguen ser automatitzades i composades per agents de programari, de manera que entenguen el contingut de les aplicacions (dades i funcionalitat) ja que estan expresades en un llenguatge que proporciona un vobulari formalitzat.

En aquest treball es proposa la generació de la part del l'especificació del sistema que serà accessible per la tecnologia de la Web Semàntica [130]. Aquesta generació es pot dur a terme ja que l'aproximació OO-Method/OOWS inclou una sèrie de models que permeten especificar de manera precisa l'estructura i comportament del sistema mitjançant un Esquema Conceptual. Ací, aquesta aproximació ha sigut enriquida amb un mecanisme per definir -a nivell de modelatge- el sistema des del punt de vista de la Web Semàntica.

Aquestes noves dimensions especifiquen l'accés/vista sobre el sistema per a agents externs, definint dos models. el model del *domini del sistema* (turisme, salut, notícies, educació, etc.) i el segon model descriu com agents i entitats externes podem emprar la funcionalitat exposada pel sistema.

10.2.5 Arquitectures Orientades a Serveis

Un objectiu principal de les *Arquitectures Orientades a Serveis* (SOA) es resoldre el problema de la integració de diferents aplicacions heterogènees en entorns distribuïts. Les arquitectures d'aquest tipus proporcionen escenaris apropiats per integrar aplicacions Web. Els mètodes d'Enginyeria Web proporcionen mecanismes per aplicar aquest tipus d'arquitectures i suportar l'ús de serveis Web externs per desenvolupar nous serveis.

D'acord a SOA, es necessari proporcionar guies metodològiques per fer un disseny i implementació automàtic de Serveis Web completament operatius a partir de models conceptuals OO-Method/OOWS. Aquestos models són considerats la part clau en aquest desenvolupament.

En aquesta estratègia, el primer que s'ha fet és determinar quins models són més útils per identificar les operacions dels Serveis Web, i després s'ha proposat una guia per desenvolupar aquestes operacions [115, 114].

Aquesta extensió és suportada per una eina que agafa models conceptuals com entrada i aplica aquesta guia per obtenir les descripcions de les operacions dels Serveis Web. Aquesta eina és la responsable finalment de generar el codi per aquestes operacions automàticament [116].

10.3 Publicacions Relacionades

Durant el periode que ha durat el desenvolupament d'aquesta tesi, s'han realitzat les següents publicacions relacionades amb el treball:

Capítols de Llibres

- **Joan Fons**, Vicente Pelechano, Oscar Pastor, Pedro Valderas, Victoria Torres. *Applying the OOWS Model-Driven Approach for Developing Web Applications. The Internet Movie Database Case Study.* **Web Engineering: Modelling and Implementing Web Applications**. Rossi G., Pastor O., Schwabe D., Olsina

L. (Eds) Springer. Human-Computer Interaction Series. ISBN: 978-1-84628-922-4. pp. 65-108. 2007

- Victoria Torres, **Joan Fons**, Vicente Pelechano. *Building Usable Business Process Driven Web Applications*. **Handbook of Research on Web Information Systems Quality**. Idea Group Inc. Calero, C., Moraga, M.A., and Piattini, M. (Eds.). (En fase de publicació) ISSN: 0950-5849. 2007

- Oscar Pastor, **Joan Fons**, Vicente Pelechano, Silvia Abrahao. *Conceptual Modelling of Web Applications: the OOWS approach*. **Web Engineering: Theory and Practice of Metrics and Measurement for Web Development**. Mendes E., Mosley N. (Eds) Springer-Verlag Berlin Heidelberg. ISBN: 3540281967. pp. 277-302. 2005. Cites: **14**

- Victoria Torres, **Joan Fons**, Vicente Pelechano. *Getting Ready Web Engineering Methods for the Semantic Web. Putting Ontologies into Practice (publicación original en ICWE 2004)*. **Engineering Advanced Web Applications**. Rinton Press. ISBN 1-58949-046-0. Volumen 9, pp. 110-123. 2004

- Vicente Pelechano, Antonio Vallecillo, Javier Muñoz, **Joan Fons**. ***Desarrollo de Software Dirigido por Modelos. MDA y Aplicaciones***. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia. ISBN: 84-688-8870-2. Pelechano V., Vallecillo A., Muñoz J., **Fons J.** (Eds). pp. 105. 2004

- Vicente Pelechano; **Joan Fons**. *Servicios Web. Aplicaciones, Tecnologías y Diseño*. **Tendencias en el Desarrollo de Aplicaciones Web**Departamento de Informática y Automática de la Universidad de Salamanca. ISBN 84-688-7872-3. pp. 85-100. 2004

- Oscar Pastor, **Joan Fons**, Vicente Pelechano. *Generación automática de aplicaciones Web a partir de Esquemas conceptuales orientados a objetos.* **Tendencias actuales en la interacción persona ordenador**: accesibilidad, adaptabilidad y nuevos paradigmas (XIII Escuela de Verano de Informática). Universidad de Castilla-La Mancha. Departamento de Informática. ISBN: 84-921873-1-X. 2003

- Vicente Pelechano, Marta Ruíz, **Joan Fons**, Pedro Valderas. *Desarrollo de Aplicaciones WEB basadas en Servicios WEB XML. Un Caso Práctico.* **Avances en Comercio Electrónico**. Francisco J. Garcia (Eds.) Avalon Programming Solutions. ISBN 84-607-5827-3. Volumen 7, pp. 99-118. 2002

- **Joan Fons**, Oscar Pastor, Pedro Valderas, Marta Ruíz. *OOWS: Un Método de Producción de Software en Ambientes Web.* **Avances en Comercio Electrónico**. Avances en Comercio Electrónico. Francisco J. Garcia (Eds.) Avalon Programming Solutions. ISBN 84-607-5827-3. Volumen 7, pp. 119-137. 2002

- Silvia Abrahao, **Joan Fons**, Magali Gonzalez, Oscar Pastor, Vicente Pelechano. *Un Método Orientado a Objetos para el Diseño de Aplicaciones de Comercio Electrónico.* **New methods and Tools Supporting E-Commerce**. pp. 125-140. 2001

Revistes

- Oscar Pastor, Silvia Abrahao, **Joan Fons**. *Building E-Commerce Applications from Object-Oriented Conceptual Models.* ACM Press, SIGecom Exchanges, Newsletter of the ACM, Special Interest Group on E-Commerce, vol. 2.2. `http://www.acm.org/sigecom/exchanges/volume_2_(01)/2.2-index.html`. pp. 28-36. 2001. Cites: **15**

Conferències Internacionals

- Francisco Valverde, Pedro Valderas, **Joan Fons**, Oscar Pastor. *An MDA-based environment for Web Application Development: From Conceptual Models to Code.* **IWWOST 2007**. International Workshop on Web Oriented Software Technologies (within the International Conference on Web Engineering, ICWE 2007). ISBN: 978-88-902405-2-2. pp. 164-178. Como (Italia), Juliol 2007

- Pedro Valderas, **Joan Fons**, Vicente Pelechano. *Developing E-Commerce Applications from Task-Based Descriptions.* **ECWEB 2005**. 6th Electronic Commerce and Web Technologies. Springer-Verlag, Lecture Notes in Computer Science, LNCS 3590. ISBN 3-540-28467-2. Copenhague, Dinamarca, August, 2005. Cites: **7**

- Pedro Valderas, **Joan Fons**, Vicente Pelechano. *From Web Requirements to Navigational Design. A Transformational Approach.* **ICWE 2005**. 5th International Conference on Web Engineering. Springer-Verlag, Lecture Notes in Computer Science, LNCS 3579. ISBN: 3-540-27996-2. pp. 506-511. Sydney, Australia. July, 2005. Cites: **3**

- Pedro Valderas, **Joan Fons**, Vicente Pelechano. *Transforming Web Requirements into Navigational Models: An MDA based Approach.* **ER 2005**. 24th International Conference on Conceptual Modelling. Springer-Verlag, Lecture Notes in Computer Science, LNCS 3716. ISBN: 3-540-28467-2. pp. 63-75. Klagenfurt, Austria. October 2005. Cites: **7**

- Gonzalo Rojas, Vicente Pelechano, **Joan Fons**. *A Model-Driven Approach to include Adaptive Navigational Techniques in Web Applications.* **IWWOST 2005**. Workshop on Web Oriented

Software Technologies (within Fifth Edition in conjunction with CAiSE 2005). Porto, Portugal, June, 2005. Cites: **11**

- Pedro Valderas, **Joan Fons**, Vicente Pelechano. *Modelling Content Aggregation for Developing E-Commerce Web Sites*. **ECWEB 2004**. 5th Electronic Commerce and Web Technologies. Springer-Verlag, Lecture Notes in Computer Science, LNCS 3182. ISBN: 3-540-22917-5, ISSN:0302-9743 September. pp. 259-267. Zaragoza, 2004

- Pedro Valderas, **Joan Fons**, Vicente Pelechano. *Extending Navigation Modeling to Support Content Aggregation in Web Sites*. **ICWE 2004**. International Conference on Web Engineering. Nora Koch, Piero Fraternalli, Martin Wirsing (Eds.). Springer-Verlag, Lecture Notes in Computer Science, LNCS 3140. ISBN: 3-540-22511-0, ISSN:0302-9743 . pp. 98-102. Munich, Germany. July, 2004

- Victoria Torres, **Joan Fons**, Oscar Asensi y Vicente Pelechano. *Getting Ready Web Engineering Methods for the Semantic Web. Putting Ontologies into Practice (publicat en "Engineering Advanced Web Applications", Ed. Rinton Press per ser un dels millors treballs enviats)*. **IWWOST 2004**. Fourth Edition of the Workshop on Web Oriented Software Technologies, in conjunction with ICWE 2004. pp. 90-100. Munich, Alemania. Julio, 2004

- Gonzalo Rojas, Vicente Pelechano, **Joan Fons**. *Navigational Properties and User Attributes for Modelling Adaptive Web Applications*. **EAW 2004**. Engineering the Adaptive Web. Workshop of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'04. ISSN: 0926-4515. pp. 52-57. Lora Aroyo, Carlo Taso (Eds.). Eindhoven, The Netherlands, August, 2004

- Victoria Torres, **Joan Fons**, Vicente Pelechano, Oscar Pastor. *Navigational Modeling and the Semantic Web. An Ontology based Approach.* LA-Web 2004. Second Latin American Web Congress. IEEE CS Press. ISBN: 0-7695-2237-8/04. pp. 94-97. *POSTER.* Brazil. October, 2004

- **Joan Fons**, Vicente Pelechano, Manoli Albert y Oscar Pastor. *Development of Web Applications from Web Enhanced Conceptual Schemas.* **ER 2003**. 22nd International Conference on Conceptual Modelling. Springer-Verlag, Lecture Notes in Computer Science, LNCS 2814. ISBN: 3-540-20299-4. pp. 232-245. Chicago, EE.UU. October, 2003. Cites: **46**

- Manoli Albert, Vicente Pelechano, **Joan Fons**, Marta Ruíz, Oscar Pastor. *Implementing UML Association, Aggregation and Composition. A Particular Interpretation based on a Multidimensional Framework.* **CAiSE 2003**. 15th International Conference on Advanced Information Systems Engineering. Springer-Verlag, Lecture Notes in Computer Science, LNCS 2681. ISBN: 3-540-40442-2. pp. 143-158. Klagenfurt, Austria. June, 2003

- Vicente Pelechano, **Joan Fons**, Manoli Albert, Oscar Pastor. *Developing Web Applications from Conceptual Models.* **CAiSE 2003**. 15th International Conference on Advanced Information Systems Engineering. Springer-Verlag, Lecture Notes in Computer Science, LNCS 2681. ISBN: 3-540-40442-2. pp. 221-224. Klagenfurt, Austria. June, 2003

- **Joan Fons**, Vicente Pelechano, Manoli Albert, Oscar Pastor, Pedro Valderas. *Extending an OO Method to Develop Web Applications using Web Services.* **WWW 2003**. Twelfth International Conference World Wide Web, *Poster Edition.* ISBN 1581136242. Budapest, Hungary. May, 2003

- **Joan Fons**, Francisco J. García, Vicente Pelechano, Oscar Pastor. *User Profiling Capabilities in OOWS*. **ICWE 2003**. Third International Conference on Web Engineering. Springer-Verlag, Lecture Notes in Computer Science, LNCS 2722. ISBN: 3-540-40522-4. pp. 486-496. Asturias, Spain. July, 2003

- Luis Olsina, Mª Angeles Martín, **Joan Fons**, Silvia Abrahao, Oscar Pastor. *Towards the Design of a Metrics Cataloging System by Exploiting Conceptual and Semantic Web Approaches*. **ICWE 2003**. Third International Conference on Web Engineering. Springer-Verlag, Lecture Notes in Computer Science, LNCS 2722. ISBN: 3-540-40522-4. pp. 324-333. Asturias, Spain. July, 2003

- Vicente Pelechano, **Joan Fons**, Manoli Albert y Oscar Pastor. *Developing Web Applications from Conceptual Models. A Web Services Approach*. **eCOMO 2003**. International Workshop on Conceptual Modeling Approaches for e-Business, 4th Edition. Dealing with Business Volatility (in conjunction with the 22nd International conference on Conceptual Modeling ER 2003). ISBN: 3-540-20257-9. Chicago, EE.UU. October, 2003

- Manoli Albert, Vicente Pelechano, **Joan Fons**, Gonzalo Rojas, Oscar Pastor. *Extracting Knowledge from Association Relationships to Build Navigational Models*. **La-Web 2003**. First Latin American Web Congress. ISBN: 0-7695-2058-8. pp. 2-10. Santiago, Chile. November, 2003

- Oscar Pastor, **Joan Fons**, Vicente Pelechano. *OOWS: A Method to Develop Web Applications from Web-Oriented Conceptual Models*. **IWWOST 2003**. Third International Workshop on Web Oriented Software Technologies. pp. 65-70. *POSTER*. Oviedo, Spain, July, 2003. Cites: **18**

- **Joan Fons**, Pedro Valderas, Oscar Pastor. *Specialization in Navigational Models*. **ICWE 2002**. International Conference on Web Engineering (withing th Conference on Computer Science and Operational Research). ISSN: 1666-6526. pp. 16-31. Santa Fe, Argentina. 2002

- Silvia Abrahao, **Joan Fons**, Magali Gonzalez, Oscar Pastor. *Conceptual Modeling of Personalized Web Applications*. **AH 2002**. 2nd International Conference on Adaptive Hypermedia and Adapative Web Based Systems. Springer, Lecture Notes in Computer Science, LNCS 2347. De Bra P., Brusilovsky P., Conejo R. (Eds.). ISBN: 3-540-43737-1, ISSN: 0302-9743. pp. 358-362. Malaga, Spain. 2002. Cites: **7**

- Magali González, Silvia Abrahao, **Joan Fons**, Oscar Pastor. *Evaluando la Calidad de Métodos para el Diseño de Aplicaciones Web*. **SBQS 2002**. I Simposio Brasilero de Qualidade de Software. ISBN: 85-88442-37-X. pp. 23-32. Gramado, Brasil. Octubre, 2002

- Oscar Pastor, Silvia Abrahao, **Joan Fons**. *An Object-Oriented Approach to Automate Web Applications Development*. **ECWEB 2001**. Second International Conference on Electronic Commerce and Web Technologies. Springer, Lecture Notes in Computer Science, LNCS 2115. ISBN: 3-540-42517-9. pp. 72-79. Munich, Alemania. September, 2001. Cites: **29**

- Pedro J. Molina, Oscar Pastor, Sofia Marti, **Joan Fons**, Emilio Insfran. *Specifying Conceptual Interface Patterns in an Object-Oriented Method with Authomatic Code Generation*. **UIDIS 2001**. Second International Workshop on User Interfaces to Data Intensive Systems. IEEE Computer Society. ISBN: 0-7695-0834-0. pp. 72-79. Zurich, Switzerland. May, 2001

Conferències Nacionals

- Ricardo Quintero, Vicente Pelechano, **Joan Fons** y Oscar Pastor. *Aplicación de MDA al Desarrollo de Aplicaciones Web en OOWS*. **JISBD 2003**. VIII Jornadas de Ingeniería del Software y Bases de Datos. Ernesto Pimentel, Nieves R. Brisaboa, Jaime Gomez (Eds.). ISBN: 84-668-3836-5. pp. 379-388. Alicante, España. Noviembre, 2003

- Silvia Abrahao, Oscar Pastor, **Joan Fons**, Luis Olsina. *Un Método para Medir el Tamaño Funcional y Evaluar la Calidad de Sitios Web*. **JISBD 2001**. VI Jornadas de Ingeniería del Software y Bases de Datos. ISBN: 84-699-6275-2. pp. 477-490. Almagro, España. Noviembre, 2001. Cites: **3**

- Silvia Abrahao, **Joan Fons**, Magali Gonzalez, Oscar Pastor, Vicente Pelechano. *Un Método Orientado a Objetos para el Diseño de Aplicaciones de Comercio Electrónico*. **ZOCO 2001**. Reunion de Trabajo sobre Metodos y Herramientas para el Desarrollo de Aplicaciones de Comercio Electronico (within JISBD?01). R. Corchuelo, A. Ruiz y José A. Pérez (Eds.). ISBN: 84-96086-02-X. pp. 125-140. Almagro, España. Noviembre, 2001

Conferències Iberoamericanes

- Francisco Valverde, Pedro Valderas, **Joan Fons**. *OOWS Suite: Un entorno de desarrollo para Aplicaciones Web basado en MDA*. **IDEAS 2007**. Workshop Iberoamericano de Ingenieria de Requisitos y Ambientes Software. Isla Margarita, Venezuela. 2007

- Ricardo Quintero, Vicente Pelechano, **Joan Fons**,Oscar Pastor. *Desarrollo de Sistemas Basados en Web mediante la aplicación de MDA a OOWS*. **CIIC 2003**. X Congreso Internacional de

Investigación en Ciencias Computacionales, CIIC 03. ISBN: 86-435-0549-8. pp. 155-162. Oaxtepec, México. Octubre, 2003

- Pedro Valderas, Marta Ruíz, **Joan Fons**, Manolo Albert, Vicente Pelechano. *Desarrollo de un portal Web para un Departamento Universitario des de Modelado Conceptual.* **IDEAS 2003**. VI Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software. Mario Piattini, Luca Cernuzzi, Francisco Ruíz (Eds.). ISBN: 84-96023-05-2. pp. 15-26. Asuncion, Paraguay. Mayo, 2003

- Mª Angeles Martín, Hernán Molina, Fernanda Papa, **Joan Fons**, Oscar Pastor, Luis Olsina. *Aspecto de Diseño Arquitectural y Semántico para un Sistema Web de Catalogación de Métricas.* **IDEAS 2003**. VI Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software. Mario Piattini, Luca Cernuzzi, Francisco Ruíz (Eds.). ISBN: 84-96023-05-2. Asuncion, Paraguay. Mayo, 2003

- Magali Gonzalez, Luca Cernucci, Silvia Abrahao, **Joan Fons**. *Aspecto de Diseño Arquitectural y Semántico para un Sistema Web de Catalogación de Métricas.* **IDEAS 2001**. IV Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software. ISBN: 959-7164-08-6. pp. 45-56. La Habana, Cuba. Abril, 2001

- Oscar Pastor, **Joan Fons**, Silvia Abrahao, Manoli Albert, Eva Campos. *Una Aproximación OO para el Modelado Conceptual de Aplicaciones Web.* **IDEAS 2001**. VIII Jornadas Iberoamericanas de Informatica: Tecnologías Software para Ambientes Web. Cartagena de Indias, Colombia. Agosto, 2001

- Oscar Pastor, **Joan Fons**, Silvia Abrahao, Sergio Ramon. *Modelado Conceptual Orientado a Objetos para Aplicaciones Web.* **IDEAS 2001**. Workshop Iberoamericano de Ingenieria de Requisitos

	Lloc	#
Capítols de Llibre	Springer (2), Rinton Press, Avalon Computing (2), UPV, ...	10
Revistes	ACM SIGecom	1
Conf. Internacionals	ICWE (5), CAiSE (3), ER (2), IWWOST (4), ECWEB (3), WWW, eCOMO, La-Web, AH (2), ...	24
Conf. Nacionals	JISBD (2), ZOCO	3
Conf. Iberoamericanes	IDEAS (5)	7
	Total	**44**

Taula 10.1 – Resum de Publicacions

y Ambientes Software. Centro de Formación Tecnologica (CIT). ISBN: 9968-32-000. pp. 239-251. San Jose, Costa Rica. Abril, 2001

10.3.1 Resum de Publicacions

La Taula 10.1 mostra informació resumida sobre les publicacions realitzades en l'àmbit de la tesi.

La següent llista mostra les publicacions més referenciades[1]:

1. **Joan Fons**, Vicente Pelechano, Manoli Albert y Oscar Pastor. *Development of Web Applications from Web Enhanced Conceptual Schemas*. **ER 2003**. Cites: **46**

2. Oscar Pastor, Silvia Abrahao, **Joan Fons**. *An Object-Oriented Approach to Automate Web Applications Development*. **ECWEB 2001**. Cites: **29**

3. Oscar Pastor, **Joan Fons**, Vicente Pelechano. *OOWS: A Method to Develop Web Applications from Web-Oriented Conceptual*

[1]Informació extreta de Google Scholar el 21-12-2007

Models. **IWWOST 2003**. Cites: **18**

4. Oscar Pastor, Silvia Abrahao, **Joan Fons**. *Building E-Commerce Applications from Object-Oriented Conceptual Models*. ACM Press, SIGecom Exchanges, Newsletter of the ACM, Special Interest Group on E-Commerce, vol. 2.2. Cites: **15**

5. Oscar Pastor, **Joan Fons**, Vicente Pelechano, Silvia Abrahao. *Conceptual Modelling of Web Applications: the OOWS approach*. **Web Engineering: Theory and Practice of Metrics and Measurement for Web Development**. Mendes E., Mosley N. (Eds) Springer-Verlag , 2005. Cites: **14**

6. Gonzalo Rojas, Vicente Pelechano, **Joan Fons**. *A Model-Driven Approach to include Adaptive Navigational Techniques in Web Applications*. **IWWOST 2005**. Cites: **11**

7. Pedro Valderas, **Joan Fons**, Vicente Pelechano. *Transforming Web Requirements into Navigational Models: An MDA based Approach*. **ER 2005**. Cites: **7**

8. Pedro Valderas, **Joan Fons**, Vicente Pelechano. *Developing E-Commerce Applications from Task-Based Descriptions*. **ECWEB 2005**. Cites: **7**

9. Silvia Abrahao, **Joan Fons**, Magali Gonzalez, Oscar Pastor. *Conceptual Modeling of Personalized Web Applications*. **AH 2002**. Cites: **7**

10. Pedro Valderas, **Joan Fons**, Vicente Pelechano. *From Web Requirements to Navigational Design. A Transformational Approach*. **ICWE 2005**. Cites: **3**

11. Silvia Abrahao, Oscar Pastor, **Joan Fons**, Luis Olsina. *Un Método para Medir el Tamaño Funcional y Evaluar la Calidad de Sitios Web*. **JISBD 2001**. Cites: **3**

Apèndix A

Framework d'Interfícies Web, WIF

A diferència dels desenvolupaments d'aplicacions tradicionals que normalment s'implementen seguint una arquitectura i llenguatge concret, en l'àmbit de la Web existeix una major hetereogenitat. En primer lloc, sempre entren en joc dos llenguatges clarament diferenciats: el llenguatge de programació o script que implementa la lògica de negoci de l'aplicació Web i el llenguatge HTML que defineix les pàgines que rep el navegador. La combinació inadequada d'ambdós llenguatge dóna lloc amb relativa facilitat a l'efecte anomenat "spaghetti" on la presentació definida en HTML es troba barrejada amb script en altre llenguatge que defineixen el seu comportament.

Aquesta problemàtica implica que les aplicacions siguen costoses de mantenir i comprendre. La introducció de noves tecnologies basades en JavaScript, com AJAX [5], no fan més que complicar més el ja de de per sí complexe panorama en el desenvolupament Web, introduint un llenguatge més al problema, que barreja inherentment aquetos conceptes. I, per si aquestos problemes no foren suficients, les incompatibilitats entre

diferentes navegadors Web a l'hora d'interpretar l'HTML de l'aplicació provoca que el desenvolupamnet Web totalment compatible siga encara més complicat, si cap.

En l'actualitat existeix una gran varietat de frameworks per al desenvolupament d'aplicacions Web. Alguns exemples que tenen una particular acceptació són Struts [126] basat en J2EE, Ruby on Rails [113], Cake PHP [22] com alternativa *OpenSource*, o el propi ASP .NET [77] de Microsoft. La proliferació d'aquest tipus de frameworks es deu a la necessitat de mecanismes que faciliten el desenvolupament d'aplicacions Web. Aquestos frameworks fan especial tractament entre la separació entre la lògica de negoci i la presentació, emprant a sovint patrons de disseny com el MVC ("*Model-View-Controller*" [100]). A més més, inclouen facilitats per desenvolupar aplicacions com ara l'*abstracció de la comunicació client-servidor*, la *simulació d'events en la interfície* o la *introducció de "*widgets*" d'interfície HTML*. L'objectiu que persegueixen no és un altre sinó fer equiparable el desenvolupament Web amb el desenvolupament tradicional.

Tanmateix, aquestos frameworks no són una solució òptima en entorns de producció de programari MDA. Encara que és cert que faciliten la implementació, les aplicacions web continuen definint-se en base al llenguatge de programació sobre el que es sustenta el framework. D'altra banda, el disseny segueix bassant-se en plantilles sobre les que s'"injecta" el codi HTML que retorna la lògica de negoci.

Des d'un punt de vista MDA (PIM/PSM-a-codi), l'avanç no és especialment significatiu. En alguns casos, inclús poden dificultar les transformacions al imposar una arquitectura que difícilment pot ser obtesa a partir dels models conceptuals. D'una banda, gran part de la lògica de negoci està lligada a conceptes d'implementació específics del framework que no té perquè estar suportat en altres frameworks. Com a conseqüència, les transformacions definides emprant un framework destí

són totalment depenents d'aquest, i no són fàcilment adaptables a altres frameworks.

Un dels principis que proposa la filosofia de les *Fàbriques de Software* [61] és la d'utilitzar frameworks que abstraguen els conceptes tecnològics relacionats a l'aplicació a construir. Seguint aquesta premisa, s'ha definit una Framework per a la construcció d'Interfícies Web (que hem anomenat WIF). Aquest framework (WIF) presenta dos diferències fonamentals respecte a la resta de frameworks actuals: (1) sols s'encarrega de definir al interfície web que es comunica amb la lògica de negoci (implementada a banda); i (2) presenta una visió més "declarativa" de la construcció d'aplicacions Web. Enlloc de sustentar-se en un llenguatge de programació, el framework es substenta en una sèrie de primitives d'alt nivell que representen conceptes habituals de les interfícies Web.

Aquest framework abstrau mitjançant un conjunt de classes, els elements comuns, com ara: pàgines, enllaços, menús, etc. D'aquesta forma, la definició d'una aplicació Web es realitza instanciant aquestes classes amb la informació del domini de l'aplicació (visualitzar informació de clients en forma de taula, crear un enllaç a la pàgina de factures, etc.). Si bé el framework ha sigut desenvolupat amb PHP 5, per a la construcció de les aplicacions només caldrà emprar la interfície d'operació ("API") que el framework proposa (veure Secció A.2.7), en base als seus objectes i operacions. D'aquesta manera, l'ús de PHP és anecdòtic, ja que els conceptes del framework podrien haver-se implementat en altre llenguatge orientat a objectes per a la web, com ara J2EE o ASP .NET (C#). Les úniques diferències entre l'ús del framework definit en un u altre llenguatge es trobarien a nivell sintàctic.

A.1 Relació d'OOWS amb el Framework

OOWS, com metodologia Web, permet definir un esquema conceptual d'una aplicació Web. Els models d'Usuaris, de Navegació i de Presentació d'OOWS descriuen a alt nivell una aplicació Web. Com s'ha vist al Capítol 8, és possible transformar aquestes descripcions en una aplicació Web concreta. Una de les solucions possibles és emprar aquest framework WIF com llenguatge objectiu en el procés de transformació.

D'altra banda, són els models definits per OO-Method (Diagrama de Classes, Model Dinàmic i Model Funcional) els que defineixen la funcionalitat del sistema. La comunicació amb aquesta lògica de negoci es produeix mitjançant una *façana de negoci* encarregada de presentar la lògica de negoci com un conjunt de serveis de recuperació d'informació i funcionalitat [25].

El framework assumeix que la informació es troba representada mitjançant un model del domini, normalment en forma d'un diagrama de classes o d'entitat-relació, de forma implícita (o per una abstracció) de la façana de negoci. Com a conseqüència, els atributs de les classes/entitats defineixen la informació emmagatzemada i les operacions i la funcionalitat ofertada.

En el cas particular de produir aplicacions Web mitjançant *OlivaNOVA* [23], la implementació comercial d'OO-Method, aquesta façana de negoci es comunica mitjançant un objecte COM+ que representa la informació mitjançant un model d'estructura (diagrama de classes). Tanmateix, el disseny del framework no obliga que aquesta lògica siga necessàriament implementada mitjançant *OlivaNOVA* i està preparat per suportar funcionalitat que provinga d'altres fonts.

A.2 WIF, Un Framework per a construir Interfícies Web

Una aplicació Web creada mitjançant el framework d'interfícies Web es fonamenta en quatre classes principals: *Application*, *Page*, *Zone* i *Service*.

Application emmagatzema l'estructura navegacional de l'aplicació web i les seues característiques globals

Page defineix una pàgina web a la que s'accedirà com a resultat de navegar per l'aplicació web

Zone especifica el contingut que forma part d'una pàgina web

Service estableix un vincle amb l'execució d'una funcionalitat de l'aplicació web

A partir d'aquestes quatre classes principals del framework és defineixen una sèrie de mètodes que permeten construir una interfície Web. A continuació es detallen les operacions que ofereix el framework, classificades en grups de funcionalitat i la classe sobre la que està definida. A més a més, es mostra el model de base a aquestes classes del framework, que forma el PSM implícit per a les transformacions de Model-a-codi.

A.2.1 Creació de l'Aplicació

El punt de partida per a la definició d'una interfície mitjançant un framework és la creació d'un objecte *Aplicació* que emmagatzeme la informació global de l'aplicació. Aquesta informació es defineix mitjançant un objecte únic ("*singleton*") de la classe *Application*. Entre altres dades, emmagatzema la referència a l'objecte amb la façana de negoci, l'usuari actualment connectat, les pàgines sobre les que aquest usuari té

permís d'accés i la seua ubicació actual en l'aplicació. Aquest objecte *Application* s'emmagatzema en la *cache de sessió* de manera que puga ser recuperat per la resta de la interfície.

A continuació es mostren les operacions específiques relacionades amb la creació d'una nova aplicació.

A.2.1.1 Application.New

Crea un objecte *Application* on se li indica el nom de la aplicació, un identificador del tipus de la lògica per defecte a emprar i l'identificador de l'objecte que encapsula la lògica de negoci. La interfície d'operació és la següent:

```
$Application = new Application(pAppName:String, pLogicType:String,
pBLId:String);
```

on:

- *pAppName* és el nom de l'aplicació Web
- *pLogicType* indica el tipus de lògica de negoci a la que es conectarà. Actualment estan suportats els següents tipus: "ONME", "WebServices" i "PHP"
- *pBLId* indica l'objecte que implementa la lògica, en funció del tipus de lògica de negoci que s'haja especificat. Aquest serà: l'identificador d'un objecte COM+ si la lògica és *OlivaNOVA*, una URL a un Servei Web o una classe *BusinessFacade* si la lògica està en PHP.

El fet de crear un objecte *Application* implica també la creació d'un objecte que representa la lògica de negoci seleccionada.

Exemple:

```
$Application = new Application(''IMDB'', ''ONME'', ''IMDBSpace'');
```

A.2.1.2 Application.DefaultStyle

Defineix l'estil CSS a aplicar a l'aplicació introduint-hi el nom del fitxer d'estils.

```
$Application->DefaultStyle(pDefStyle:String);
```

Aquest estil ha de ser un dels que existeixca al recurs "styles" de l'aplicació web.

Extemple:

```
$Application->DefaultStyle(''IMDB'');
```

A.2.2 Autenticació d'Usuaris

El framework incorpora un mecanisme d'autenticació d'usuaris basat en *Rols*. Un Rol representa a un grup d'usuaris que comparteixen els permissos, podent establir herència entre rols (els rols fills hereten els permissos del rol pare). El framework presuposa que la lògica de negoci proporciona un servei per autenticar usuaris i retorna un valor lògic (booleà). Els mètodes que configuren l'autenticació d'usuaris són: Application.Rol, Application.AllowAnonymous.

A.2.2.1 Application.Rol

Un Rol defineix a un grup d'usuaris de l'aplicació sobre el que se'ls otorga permís per accedir o no a una pàgina determinada. Un rol es defineix en base a un identificador únic, un àlies, un servei de la lògica de negoci d'autenticació i un rol pare. L'aplicació espera que un usuari s'identifique mitjançant un *login* i un *password* que seran enviats a aquest servei

d'autenticació. Si el servei retorna "cert", es permetrà a l'usuari entrar n l'aplicació. D'altra banda, el rol pare permet definir un mecanisme d'herència entre rols de manera que el fill herete els permisos d'accés a l'aplicació del rol pare.

```
$Application->Rol(pRol:String, pRolAlias:Sring, pAutService:String,
pParentRol:String);
```

Exemple: El següent exemple defineix un Rol "Logged Client" que hereta els permisos de l'usuari anònim i realitza l'autenticació amb el servei "Client_Login". A més a més, es defineix un Rol "Administrador" que hereta del rol anterior i per tant també dels usuaris anònims.

```
$Application->Rol(''Logged Client'', ''Client'', ''Client_Login'',
''Anonymous'');
$Application->Rol(''Administrator'', ''Administrator'',
''Administrator_Login'', ''Logged Client'');
```

A.2.2.2 Application.AllowAnonymous

Habilita la possibilitat de que puguen accedir usuaris anònims a l'aplicació.

```
$Application->AllowAnonymous();
```

A.2.3 Definició de les Pàgines Web

En una interfície Web, les pàgines són l'element principal, ja que actuen com els contenidors de la informació, la funcionalitat i permeten a l'usuari interactuar amb el sistema. La missió de la classe *Page* del framework és la de representar una pàgina Web. Cada pàgina és creada en un fitxer individual amb el nom de l'identificador de la pàgina, i

estableix la URL d'accés. El conjunt de pàgines que formen la interfície Web és definit a l'objecte *Application*, creant referències a les diferents pàgines i establint el rol que té accés a cadascuna, junt amb la seua *accessibilitat*.

El framework també permet la possiblitat de definir grups de pàgines que divideixen la interfície en subsistemes per facilitar la navegació. A partir de tota aquesta informació el framework genera de forma automàtica un menú de navegació principal (*zona de navegació*) que comparteixen totes les pàgines de la interfíce. Aquest menú esà format per totes les pàgines definides amb accessibilitat "*Always*" que pertanyen al rol de l'usuari connectat. Els mètodes disponibles per a la creació de pàgines són: *Application.Page*, *Application.PageGroup*, *Application.AddPageToGroup*, *Application.HomePage*, *Page.New*.

A.2.3.1 Application.Page

Afig la referència a una pàgina a l'objecte *Application*. Rep l'identificador de la pàgina que deu coincidir amb l'identificador (nom del fitxer) de l'objecte *Page* al que repesenta, un àlies, el rol que té permís per accedir a la pàgina i l'accessibilitat. Aquest últim paràmetre pot prendre únicament dos valors: *Always*, si la pàgina pot ser accedidad des d'enlloc en l'aplicació web, o *FromPage* si sols es possible accedir a través d'una seqüència de navegació concreta (a través d'un enllaç explícit en una altra pàgina).

Exemple: Es mostra la definició de dues pàgines, *MovieComments* i *MovieInformation* amb diferents rols i accessibilitat:

```
 $Application->Page(''MovieComments'', ''Comments'', ''Registered User'',
''Always'');
$Applciation->Page(''MovieInformation'', ''Movie Detail'', ''Anonymous'',
''FromPage'');
```

La primera pàgina, *MovieComments*, definida amb l'àlies *Comments*, està accessible des de qualsevol pàgina per al rol *Registered User*.

La segona pàgina, *MovieInformation*, definida amb l'àlies *Movie Detail*, està accessible per als usuaris anònims però sols a partir d'altres pàgines concretes[1].

A.2.3.2 Application.PageGroup

El framework permet la definició d'agrupacions de pàgines a fi de facilitar la navegació. Un grup rep un identificar que fa referència a un objecte *Page* que representa la pàgina inicial del grup, un rol que tindrà permisos sobre totes les pàgines del grup i un grup pare si el grup es troba dins un altre grup (subgrup). Aquest últim paràmetre rep el valor *Root* si no es troba dins de cap altre grup.

Exemple:

```
$Application->PageGroup(''MoviesBrowse'', ''Anonymous'', ''Root'');
```

A.2.3.3 Application.AddPageToGroup

Mitjançant aquest mètode és possible afegir referències a pàgines dins d'un grup, de manera similar al mètode *Application.Page*. La única diferència és que el rol no és necessari definir-lo, ja que ja ha sigut definit al crear al grup.

Exemple:

```
$Application->AddPageToGroup(''MoviesGenre'', ''ByGenre'', ''MoviesBrowe'',
''Always'');
```

[1] Els enllaços a aquesta pàgina es definiran mitjançant el mètode *Field.InternLinkTo* a la Secció A.2.7.3

A.2.3.4 Application.HomePage

Aquest mètode permet establir quina serà la pàgina d'inici d'una aplicació. Deu verificar-se que els usuaris tinguen un rol adequat per poder accedir a aquesta pàgina, ja que en cas contrari no podran accedir a l'aplicació.

Exemple:

```
$Application->HomePage(“TopMovies”);
```

A.2.3.5 Page.New

Inicialitza un objecte de tipus pàgina (*Page*) a partir del qual es podran afegir les diferents zones de contingut. Rep l'àlies de la pàgina, que és el que serà visible en l'aplicació Web.

Exemple:

```
$Page = new Page(“Top Movies”);
```

A.2.4 Recuperació de la Informació

Si tenim en compte que fonamentalment la funció d'una aplicació Web és la de mostrar informació, comprendrem la vital importància d'aquesta característica. La recuperació d'informació es representa al framework mitjançant zones d'informació. Aquestes zones fan referència a classes, atributs i relacions del model del domini que són les encarregades d'emmagatzemar la informació de manera estructurada.

Una *Zona d'Informació* s'encarrega tant de recuperar la informació solicitada des de la lògica de negoci, com de presentar-la a l'usuari de manera adequada. Les operacions que permeten definir les zones d'in-

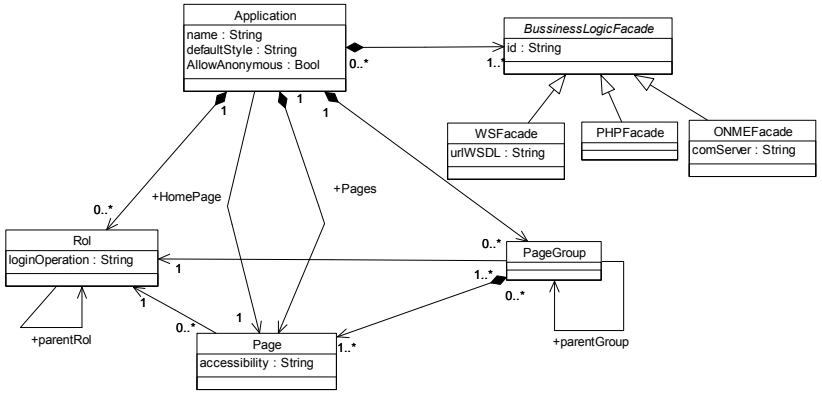


Figura A.1 – Model de l'aplicació, autenticació i definició de pàgines

formació són: *Page.AddInformationZone*, *InformationZone.Field*, *InformationZone.AddRelatedField* i*InformationZone.DetailRelationship*.

A.2.4.1 Page.AddInformationZone

Afig una zona d'informació que recupera les instàncies associades a una classe de la lògica de negoci (classe principal). Una zona d'informació es defineix mitjançant un identificador, un àlies, el nom de la classe principal i una variable lògica que indica si la zona d'informació té en compte un filtre navegacional o no. Si com a conseqüència d'una navegació, la pàgina rep una instància d'objecte, la zona d'informació únicament recuperarà la informació associada a aquesta instància.

Exemple: En el següent exemple es recuperen totes les instàncies de la classe *Movie*. Si la pàgina rebera un identificador d'una instància d'una pel·lícula, sols es recupearia la informació associada a aquesta instància.

```
$InfoZone = $Page->AddInformation(‘‘TopMovies’’, ‘‘Top Movies’’, ‘‘Movie’’,
true);
```

A.2.4.2 InformationZone.Field

Afig un nou camp (atribut) per a ser mostrat en una zona d'informació. El camp estarà relacionat amb l'identificador d'un atribut de la classe principal que forma la zona d'informació. A més a més, podrà definir-se opcionalment un àlies. Per accedir a l'atribut i utilitzar els seus mètodes, s'empra la següent notació:

```
$zona->idAtribut->mètode
```

Exemple: El següent exemple mostra la informació relativa als atributs *rating* i *commentsNumber* que pertanyen a la classe *Movie*.

```
$InfoZone->Field(''rating'', ''Rating'');
$InfoZone->Field(''commentsNumber'', ''Votes'');
```

A.2.4.3 InformationZone.AddRelatedField

Afig un camp (atribut) d'una classe relacionada de manera univaluada[2] amb la classe principal de la zona d'informació. És a dir, cada instància de la zona d'informació tindrà un únic valor per a aquest camp. El mètode rep l'identificador de l'atribut, l'àlies amb el que es mostrarà i l'identificador de la classe relacionada, a la que pertany.

Per accedir a l'atribut i utilitzar els seus mètodes s'empra la següent notació:

```
$zona->RolClasseRelacionada_idAtribut->mètode
```

[2]Existeix una relació estructural en el diagrama de classes amb extrem "a-un" amb respecte a la classe origen.

Exemple: Suposant que la classe principal de la zona té una relació a-un amb la classe *AgeRating*, per cada instància recuperada d'una pel·lícula podria veures la seua classificació per edad:

```
$Zone->RelatedField(''value'', ''Age rating'', ''AgeRating'');
```

Nota: En l'estat actual d'implementació del framework sols es suporta definir camps relacionats directament amb la classe principal, encara que es planteja donar solució completa i permetre definir recuperacions a partir de classes complementàries també.

A.2.4.4 InformationZone.DetailRelationship

Crea una nova pàgina d'informació de detall a partir d'una zona d'informació principal o mestra. Un detall representa un conjunt d'instàncies relacionades amb una altra instància de la classe principal. Per tant, la classe que defineix la zona d'informació de detall deu tenir una relació multivaluada amb la classe principal de la zona d'informació.

Una vegada definit el detall, aquest es comporta de manera similar a una zona d'informació en la que la classe relacionada (detall), actua com la classe principal de la seua zona d'informació. Per tant, és possible afegir camps d'informació, mecanismes d'accés i inclús altres zones, recursivament.

Una *zona de detall* es defineix a partir de l'identificador de la classe detall, l'identificador de la relació i l'àlies per a la zona d'informació de detall. L'identificador de la relació, junt amb la classe relacionada és la informació que se li passa a la lògica de negoci per poder recuperar la informació relacionada. Per tant, en certa manera, es depenent d'aquesta lògica. Per al cas concret d'emprar la lògica de negoci amb *OlivaNOVA*, aquest identificador de la relació es forma mitjançant la classe relacionada, un guió baix (_) de separació i el nom del rol de la

relació entre ambdues classes. Es pot apreciar que aquesta aproximació és allò suficientment genèrica sempre i quant el diagrama de classes que representa l'estructura d'informació empre els rols, tal com els defineix l'UML.

Exemple: Es defineix un detall per a les instàncies dels gèneres de pel·lícules (*Genre*), de manera que mostre totes les pel·lícules associades a cada gènere. La classe relacionada és per tant *Movie*, i l'identificador de la relació es forma mitjançant la classe oposada en la relació, en aquest cas *Genre* junt amb el rol definit en el diagrama de classes entre ambdues classes (en aquest cas, també *Genre*). De cada pel·lícula es mostrarà el seu títol i l'any en què s'estrenà.

```
$DetailMovie = $Zone->DetailRelationship(‘‘Movie’’, ‘‘Genre_Genre’’,
‘‘Genres’’);
$DetailMovie->Field(‘‘title’’, ‘‘Title’’);
$DetailMovie->Field(‘‘year’’, ‘‘Year’’);
```

A.2.5 Presentació de la Informació

Per defecte, les instàncies d'una zona d'informació es mostren a l'usuari sense cap tipus d'estructura. Tanmateix, és habitual que certs atributs tinguen una visualització distinta a la textual o que calga limitar el número d'instàncies. Els mecanismes de presentació s'encarreguen d'aquestes tasques. Cal destacar que aquestos mecanismes no influeixen sobre l'aparença més visual de la informació (color, tamany de font, posició, etc.) sinó més bé en la seua estructura de presentació amb relació a altres elementes.

Els mètodes disponibles són els següents: *Field.Type*, *InformationZone.AddSortCondition*, *InformationZone.SetPagination*, *InformationZone.SetMaxInstances* i *InformationZone.SetLayout*.

A.2.5.1 Field.Type

A través de la propietat *Type* és possible determinar com la informació que conté un camp serà visualitzada a l'hora de tornar la pàgina Web. En altres paraules, defineix el tipus de presentació del camp. Els tipus possibles, en la implementació actual, són:

- *Text:* Per defecte, la informació recuperada en un camp es mostra a l'usuari de forma textual, és a dir, com una simple cadena.
- *Number:* La informació es segueix mostrant de manera textual, però l'identificador de l'estil de camp varia per poder establir un estil diferenciat al de la informació textual.
- *Decimal:* Defineix un número amb dues possicions decimals a partir d'un camp numèric. A més a més, afig un identificador d'estil diferent.
- *Imatge:* Mostra el camp com una image mitjançant l'etiqueta IMG (de HTML). El camp deurà contenir una URL vàlida que apunte a la image.
- *ExternLink:* Mostra el camp com un enllaç mitjançant l'etiqueta A (de HTML). Aquest tipus d'enllaç únicament deu emprar-se per a enllaços externs a l'aplicació.
- *MailTo:* Estableix el camp com un link a una adreça de correu electrònic mitjançant l'identificador *mailto* en la URL de l'etiqueta A (de HTML). El camp ha de contenir una adreça de correu electrònica vàlida.
- *Multimedia:* Defineix la visualització del camp com un objecte multimedia (generalment un arxiu flash), mitjançant l'etiqueta EMBED (de HTML).

Per accedir a aquest atribut, s'empra la següent notació, depenent d'on es trobe l'atribut. Per a camps de la classe principal:

```
$zona->idAtribut->Type(pTipus)
```

Per a camps de classes relacionades:

```
$zona->ClasseRelacionada_idAtribut->Type(pTipus)
```

Exemple:

```
$InfoZone->rating->Type(“Decimal”);
```

A.2.5.2 InformationZone.AddSortCondition

Estableix una condició d'ordenació sobre algun dels camps definits en la zona de informació. Aquest mètode rep l'identificador de l'atribut i el sentit de la ordenació (ASCendent o DESCendent).

Exemple:

```
$InfoZone->AddSortCondition(“rating”, “DESC”);
```

A.2.5.3 InformationZone.SetPagination

Defineix el número d'instàncies que formen una pàgina en la zona d'informació. Aquesta primitiva activa automàticament el mecanisme de paginació de tal manera que l'aplicació mostra un únic conjunt d'instàncies de cardinalitat definida. L'usuari podrà navegar a la pàgina anterior o posterior, així com directament a la pàgina en concret a través d'un número de posició.

Nota: Com es pot observar, la implementació actual només dóna suport a la paginació estàtica. És a dir, no permet canviar-la dinàmica-

ment. És una tasca pendent al framework permetre aquesta paginació dinàmica.

Exemple:

```
$InfoZone->SetPagination(50);
```

A.2.5.4 InformationZone.SetMaxInstances

Estableix el número màxim d'instàncies a mostrar en una zona d'informació (per temes de rendiment).

Exemple:

```
$InfoZone->SetMaxInstances(1000);
```

A.2.5.5 InformationZone.SetLayout

Defineix com s'estructura la informació de les instàncies recuperades. Actualment existeixen tres tipus possibles de distribucions de presentació (*layouts*) que poden ser definides, més un quart tipus (Mestre-Detall) que s'empra automàticament quan es defineixen detalls en la zona d'informació.

- *TabularVertical:* és la distribució per defecte. Les instàncies són mostrades en forma de taula on la capçalera de les columnes són els àlies dels diferents camps de la taula. Cada fila addicional representa una instància.
- *TabularHoritzontal:* les instàncies es mostren en una composició tabular, però en aquest cas les files de la primera columna contenen els àlies dels camps. Cada columna addicional representa una instància.

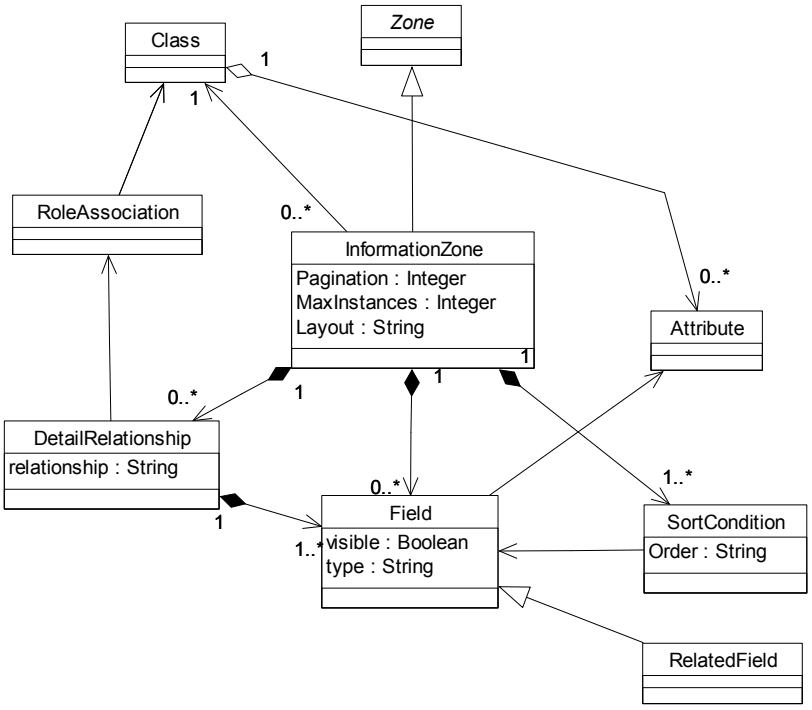


Figura A.2 – Model per a l'Especificacio de la Informació

- *Registre:* es mostra tota la informació associada a una instància (àlies i valors dels seus camps) un a un.

Exemple:

```
$InfoZone->SetLayout(‘‘Register’’);
```

A.2.6 Mecanismes d'accés a la informació

Sempre que l'usuari accedeix a una zona d'informació, són recuperades totes les instàncies que formen part de la classe principal i aquelles relacionades mitjançant detalls. En molts casos, aquest comportament pot comprometre la usabilitat ja que l'usuari pot verure's desbordat per la quantitat d'informació recuperada. Els mecanismes d'accés a la informació que ofereix el framework pretenen solucionar aquest problema restringint el conjunt d'instàncies que són recuperades des de la lògica de negoci.

El framework suporta dos tipus de mecanismes d'accés: (1) Índexs, que estableixen particions en el conjunt d'instàncies a partir dels valors que pot prendre un atribut (creant classes d'equivalència), i (2) Filtres, que defineixen una condició lògica sobre un atribut de manera que sols es poden recuperar aquelles instàncies que la compleixen.

Opcionalment, per a ambdós mecanismes és possible definir una vista de búsqueda o de resultats (*Search View*) que mostre de forma abreviada (utilitzant uns pocs atributs) el conjunt d'instàncies recuperades a l'aplicar el mecanisme d'accés.

Els mètodes que poden emprar-se per la definició d'aquestos mecanismes són: *InformationZone.DefineIndex*, *InformationZone.DefineStaticFilter*, *InformationZone.DefineDynamicFilter*, *Filter/Index.SearchView*, *SearchView.Attribute* i *SearchView.SelectionAttribute*.

A.2.6.1 InformationZone.DefineIndex

Defineix un índex en una zona d'informació sobre un atribut de la classe principal o d'una classe relacionada amb aquesta directament. Rep com paràmetres l'identificador de l'índex, un àlies i l'identificador de l'atribut (en la forma: "nom de la classe + . + nom de l'atribut") que actua com indexador.

Exemple: Suposant una zona d'informació definida sobre la classe

gèneres de pel·lícules (*Genre*), el següent índex crearia una llista amb tots els gèneres disponibles:

```
$i_Genre = $InfoZone->DefineixIndex(‘‘i_Genre’’, ‘‘Select Genre’’,
‘‘Genre.value’’);
```

A.2.6.2 InformationZone.DefineStaticFilter

Crea un filtre de caràcter estàtic sobre un atribut de la classe principal o directament relacionat amb aquesta. Un filtre estàtic és aquell en que la condició de filtrat està format per una fòrmula lògica tancada (sense paràmetres). L'usuari podrà activar o desactivar el filtre, però no podrà modificar la fòrmula de filtrat predefinida en temps de disseny.

Aquesta operació rep com argument l'identificador del filtre estàtic, un àlies per al filtre, l'atribut de la classe sobre la que actua el filtre, un operador de filtre i el valor de la fòrmula que deu complir l'atribut.

Els operadors de filtre actualment disponsibles són: “Equal” (els dos valors són idèntics), “Aproximate” (la cadena alfanumèrica del valor del filtre està continguda en el valor de l'atribut), “Greater” (major estricte), “GreaterEqual” (major o igual), “Lesser” (menor estricte) i “LesserEqual” (menor o igual).

Exemple: Suposant una zona d'informació definida sobre la classe *Movie*, el següent filtre mostra les películes produides a partir de l'any 2000.

```
$f_y2000 = $InfoZone->DefineStaticFilter(‘‘f_y2000’’, ‘‘Movies from year
2000’’, ‘‘Movie.year’’, ‘‘GreaterEqual’’, ‘‘2000’’);
```

A.2.6.3 InformationZone.DefineDynamicFilter

Defineix un filtre de caràcter dinàmic sobre un atribut de la classe principal o directament relacionada amb aquesta. A diferència del filtre

estàtic, en un filtre dinàmic la condició de filtrat està formada per una fòrmula lògica oberta, és a dir, amb paràmetres. Com a conseqüència, l'usuari deurà introduir un valor sobre el qual s'aplique la condició de filtrat.

Un filtre dinàmic es defineix mitjançant un identificador, un àlies, l'identificador de l'atribut de la classe principal i un operador de filtre (igual que els dels filtres estàtics).

Exemple: El següent filtre rep un any introduit per l'usuari i retorna les pel·lícules produides en aqueix any.

```
$f_year = $InfoZone->DefineDynamicFilter(''f_yar'', ''Movies by year'',
''Movie.year'', ''Equal'');
```

A.2.6.4 Filter/Index.SearchView

Tant als filtres com als índex se'ls pot associar una vista de resultats (*Search View*) que mostre els resultats d'executar el mecanisme. La vista es composa d'un conjunt d'atributs de la classe principal i almenys un atribut de selecció, que al ser seleccionat per l'usuari mostra tota la informació de la instància.

Una vista de resultat es defineix a partir de la classe a la que fa referència, i un àlies. Es deu tenir un compte, a efectes pràctics, que una vista de resultats es comporta com una zona d'informació.

Exemple: El següent codi defineix una vista de resultats sobre la classe *Actor* associat al filtre "f_fActor".

```
$f_fActor->SearchView(''Actor'', ''Actor names'');
```

A.2.6.5 SearchView.Attribute

Afig un atribut per ser mostrat en una vista de resultats. Els atributs poden pertanyer tant a la classe principal de la vista com a una classe relacionada de forma directa i amb cardinalitat univaluada.

Rep com paràmetres l'identificador de l'atribut i un àlies.

Exemple: S'afegeix l'atribut "Actor.first_name" i "Actor.last_name" a la vista de resultats.

```
$f_fActor->Attribute(''Actor.first_name'', ''First Name'');
$f_fActor->Attribute(''Actor.last_name'', ''Last Name'');
```

A.2.6.6 SearchView.SelectionAttribute

Defineix l'atribut de selecció en una vista de resultats. L'atribut deu pertànyer a la classe principal sobre la que es defineix la vista. Provoca que aquest atribut siga visible en la vista de resultats (si no ho era ja)

Rep com paràmetres l'identificador de l'atribut i un àlies.

Exemple: Es defineix l'atribut "Actor.last_name" com l'atribut de selecció sobre el que s'aplicarà l'enllaç.

```
$f_fActor->SelectionAttribute(''Actor.last_name'', ''Last Name'');
```

A.2.7 Especificació de la Navegació

En el domini de les aplicacions Web la interacció fonamental és la navegació. El framework permet definir tant la navegació tradicional, entesa com la transició d'una pàgina a una altra, com la navegació objectual. Aquesta interacció de navegació està associada a l'exploració de l'estructura d'informació del sistema (segons el diagrama de classes), a base d'anar seleccionant instàncies d'objectes i obtenint la seua informació relacionada. El framework també contempla la navegació per relació,

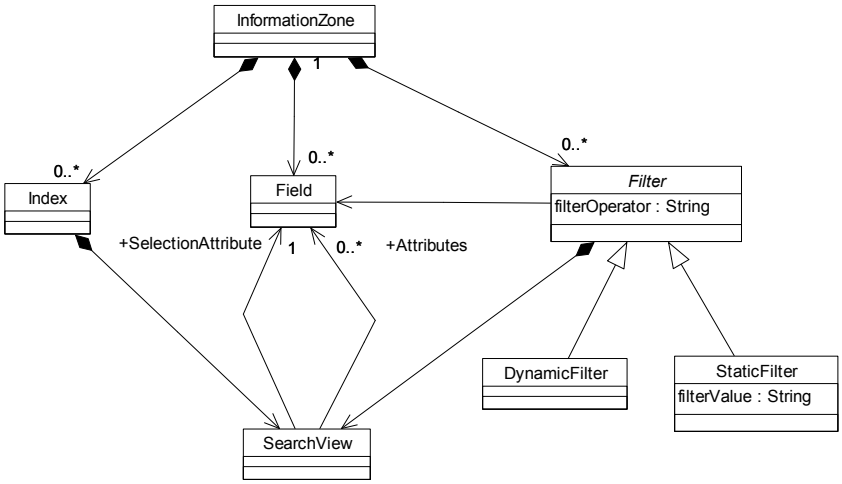


Figura A.3 – Model per a la Definició de Mecanismes d'Accés

que és aquella en la que enlloc de seleccionar una instància d'un objecte es selecciona una relació d'un objecte, per tal de veure tota la informació relacionada amb aqueixa relació.

Ja que les instàncies dels objectes són representats en el framework mitjançant zones d'informació, aquest tipus de navegació s'associa a elements d'aquesta zona. El framework incorpora un mecanisme de filtrat objectual (activat a l'hora de crear la zona d'informació) que permet que una zona d'informació sols mostre la informació relacionada amb l'identificador de l'objecte (i de la relació) rebut a causa de la navegació. Aquest comportament de filtrat el fa de manera automàtica el framework.

Les operacions que suporten la navegació són: *Page.AddNavigation*, *NavigationZone.NavigationLink*, *Field.InternLinkTo* i *InformationZone.RelationshipLinkTo*.

A.2.7.1 Page.AddNavigationZone

Afig una zona de navegació que està formada per un menú amb diferents enllaços de navegació. Per defecte, tota pàgina té una zona de navegació amb els enllaços a les pàgines amb accessibilitat *"Always"* .

Exemple:

```
$NavZone = $Page->AddnavigationZone(''MainMovies'', ''Goto Main Movies'');
```

A.2.7.2 NavigationZone.NavigationLink

Aquest mètode actua sobre una zona de navegació i permet afegir un enllaç per navegar a una altra pàgina de l'aplicació. Rep l'identificador de la pàgina destí, un àlies per a l'enllaç i el nivell d'anidament en el menú. Aquest nivell permet definir menús amb varios nivell, sent el primer nivell *"Root"* .

Exemple: En el següent exemple, la zona de navegació estaria formada per dos enllaços, on l'enllaç *ActionMovies* es trobaria en un segon nivell d'anidament dins *TopMovies*.

```
$NavZone->NavigationLink(''TopMovies'', ''Goto Movies'', ''Root'');
$NavZone->NavigationLink(''ActionMovies'', ''Action'', ''Top Movies'');
```

A.2.7.3 Field.InternLinkTo

Aquest mètode defineix un enllaç de navegació objectual associat a un camp de la zona d'informació. Quan l'usuari seleccione aquest camp, es navegarà a la pàgina destí de l'aplicació que rebrà l'identificador que activà la navegació.

Exemple: En aquest exemple, quan l'usuari seleccione el títol de la pel·lícula es navegarà a la pàgina *MovieInformation*, enviant l'identificador (oid) de la pel·lícula seleccionada.

```
$InfoZone->title->InternLinkTo(‘‘MovieInformation’’);
```

A.2.7.4 InformationZone.RelationshipLinkTo

De manera similar a *Field.InternLinkTo*, aquesta operació defineix un enllaç de navegació, però associat a una relació enlloc de a un objecte. Per a cada instància de la classe principal s'afig un camp addicional que representa un enllaç que activa aquest tipus de navegació.

Aquest mètode rep l'identificador de la relació (definit de la mateixa manera que en *DetailRelationship*), un àlies per a l'enllaç que serà mostrat i la pàgina destí. des de la pàgina destí es tindrà visibilitat tant de l'identificador de l'objecte seleccionat com l'identificador de la relació.

Exemple: El següent enllaç definit sobre la classe *Movie* mostraria un enllaç *Cast* per cada instància. Quan l'usuari seleccione l'enllaç, navegarà a la pàgina *ActorInformation* on veurà tots els actors relacionats (com a participants) en la pel·lícula seleccionada.

```
$InfoZone->RelationshipLinkTo(‘‘Movie_Movie’’, ‘‘Cast’’,
‘‘ActorInformation’’);
```

A.2.8 Serveis. Execució de la funcionalitat

Si la classe *Page* i les *zones d'informació* són els elements fonamentals a l'hora de definir la informació de la interfície, la classe *Service* i les *zones de servei* fan aquest paper quan es necessari fer disponible a l'usuari la funcionalitat del sistema.

Un *servei* representa una interfície (funcional), típicament un formulari d'entrada de dades que causa l'execució d'algun mètode ofertat per la lògica de negoci. A l'igual que amb les pàgines, un servei es defineix

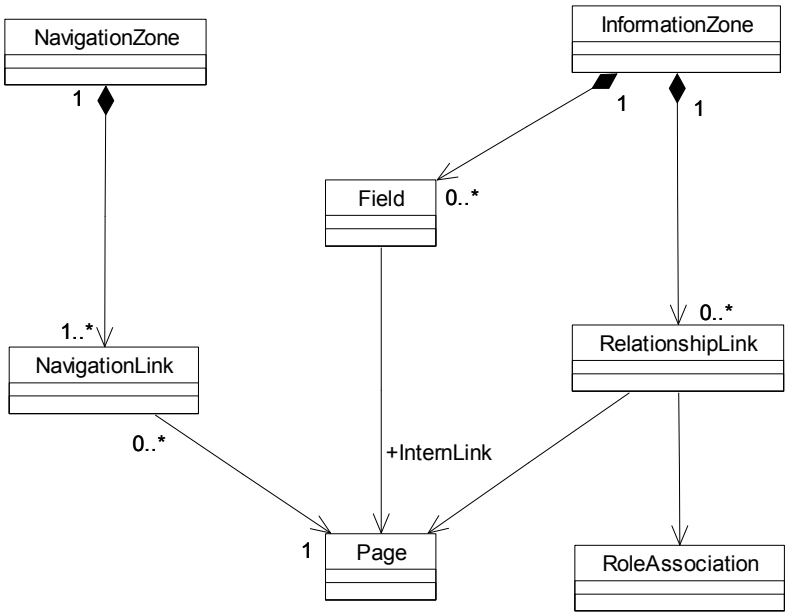


Figura A.4 – Model de la Navegació

en un fitxer propi amb un nom que comparteix amb l'identificador del servei a la lògica de negoci i que defineix la URL d'accés.

Dos tipus de zones de contingut s'empren a l'hora de definir un servei: (1) les zones del servei, formades per enllaços que naveguen al formulari del servei, i (2) les zones d'entrada de dades, definides de manera implícita al crear un servei, on s'especifiquen els camps del formulari en base als arguments del servei.

Les operacions que ofereix el framework són les següents: *Page.AddServicesZone*, *ServiceZone.ServiceReference*, *Service.New*, *Argument.Value*, *Service.Arg<TipusArgument>*, *Argument.NotRequired*, *Argument.NotVisible*, *NumberArgument.SetNatural*, *TextArgument.Lines*, *Argument.SetStaticValues* i *Argument.SetDynamicValues*.

A.2.8.1 Page.AddServicesZone

Defineix una zona en la que es mostren una sèrie d'enllaços que activen l'execució d'un servei o naveguen cap al formulari d'entrada dels paràmetres.

Rep com paràmetres un identificador i un àlies per a la zona.

Exemple:

```
$ServZone = $Page->AddServicesZone(‘‘MovieServicesZone’’, ‘‘Movie
Services’’);
```

A.2.8.2 ServiceZone.ServiceReference

Afig un nou enllaç en una zona de serveis que fa referència al formulari d'execució del servei.

Rep com arguments l'identificador del servei i un àlies per a l'enllaç.

Exemple:

```
$ServZone->ServiceReference(‘‘Movie_CREATE_MOVIE’’, ‘‘New’’);
```

A.2.8.3 Service.New

Crea un nou objecte servei que abstrau un formulari per a l'execució d'una operació de la lògica de negoci. Cada objecte servei deu ser creat en un fitxer de configuració individual, amb el mateix nom que l'identificador del servei.

Únicament rep com argument l'identificador l'àlias que descriurà el formulari.

Exemple:

```
$Service = new Service(‘‘New’’);
```

A.2.8.4 Service.Arg<TipusArgument>

Afig un argument del tipus indicat a un objecte servei. Cada argument introduit serà mostrat a l'usuari com un camp junt a una etiqueta descriptiva. Tots els mètodes reben com argument un identificador de l'argument i un àlies. Els tipus d'arguments que poden afegir-se són els següents:

- *ArgText:* defineix un argument de tipus text (cadena de caràcters). Rep com paràmetre addicional el número de caracters (longitud) del camp.
- *ArgNumber:* defineix un camp per a la introducció d'un valor numèric no decimal.
- *ArgReference:* defineix un argument de tipus referència a objecte. Utilitza un tercer paràmetre que indica la classe a la que pertany l'objecte. El camp que es crea en el formulari és de tipus numèric en el que s'espera que s'introduisca un identificador d'objecte vàlid.
- *ArgBool:* afig un argument de tipus lògic. Típicament es representa com un *checkbox*.
- *ArgTime:* crea un camp per a la introducció d'un argument de tipus "temps" (hores i minuts).
- *ArgDate:* afig un camp de tipus data que permet introduir el dia, el mes i l'any. Valida que la data és correcta.
- *ArgReal:* defineix un camp per a la introducció d'un valor numèric en el domini dels números reals.
- *ArgPassword:* crea un camp de tipus contrassenya en la que allò que s'introdueix roman ocult. L'argument és de tipus textual.

Una vegada definits els arguments del servei, és possible accedir als mètodes que aquestos proporcionen mitjançant la sintaxi:

```
$idZonaServei->idArgument
```

.

Exemple: Al següent codi es defineix un servei que rep un títol (màxim 32 caracters), un any i un objecte de la classe director.

```
$Service->ArgText(''pt_p_atrtitle'', ''Title'', 32);
$Service->ArgNumber(''pt_p_atryear'', ''Release Year'');
$Service->ArgReference(''pt_p_evcDirector'', ''Director'', ''Director'');
```

A.2.8.5 Argument.Value

Permet definir el valor per defecte que rep un atribut.

Rep com paràmetre un valor que haurà de ser del mateix tipus que l'argument.

Exemple: Inicialitza l'argument *Release Year* a 2007.

```
$Service->pt_p_atryear->Value(2007);
```

A.2.8.6 Argument.NotRequired

Indica que l'argument és opcional per a l'execució del servei i per tant no és necessari que l'usuari l'introduisca. Per defecte, tots els arguments són necessaris.

Exemple:

```
$Service->pt_p_plot->NotRequired();
```

A.2.8.7 Argument.NotVisible

Oculta el camp d'un argument de manera que l'usuari no puga introduir cap valor. Sol deu emprar-se si l'argument és opcional o s'ha definit un valor per defecte. Tots els arguments són visibles a menys que s'indique el contrari.

Exemple:

```
$Service->pt_p_atryear->NotVisible();
```

A.2.8.8 NumberArgument.SetNatural

Estableix un argument numèric com natural de manera que no pot rebre valors negatius.

Exemple:

```
$Service->pt_p_atryear->SetNatural();
```

A.2.8.9 TextArgument.Lines

Permet definir un argument textual compost per varies línies de text d'una longitud introduïda. Rep com paràmetre el número de línies.

Exemple:

```
$Service->ArgText(‘‘pt_p_atrplot’’, ‘‘Plot_Outline’’, 32);
$Service->pt_p_atrplot->Lines(4);
```

A.2.8.10 Argument.SetStaticValues

Estableix un conjunt de valors possibles per a un argument per que l'usuari en seleccione un a través d'un seleccionat (típicament una llista desplegable). Aquest conjunt estarà format per un vector de valors que

rep el mètode com argument. Aquestos valors deuen ser compatibles amb el tipus definit per l'argument al que estan associats.

Exemple: El següent codi estableix que l'usuari sols podrà seleccionar valors compresos entre el 0 i el 5 per a l'atribut *p_ atrrate*.

```
$Service->p_atrrate->SetStaticValues(array(0,1,2,3,4,5));
```

A.2.8.11 Argument.SetDynamicValues

Estableix un conjunt de valors possibles per a un argument basant-se en una població de l'estructura d'informació del sistema (atribut d'una classe). Un conjunt dinàmic es defineix a partir de l'identificador d'una classe, un atribut de la classe que defineix els possibles valors de l'argument i un altre atribut que defineix el text que serà mostrat en el selector (típicament una llista desplegable).

Aquest mecanisme millora la usabilitat, ja que permet mostrar a l'usuari un valor més descriptiu per a la seua selecció, mentre que al servei se li envia el valor de l'atribut.

Exemple: El següent exemple defineix per a l'argument de referència *p_ agrMovie*, un conjunt format per tots els identificadors d'objecte de la classe *Movie* que seleccionarà l'usuari mitjançant l'atribut *title* corresponent.

```
$Service->p_agrMovie->SetDynamicValues(‘‘Movie’’, ‘‘id_Movie’’, ‘‘title’’);
```

A.2.9 Definició d'altres zones de contingut

Les zones d'informació, navegació i servei engloben la major part de la definició d'una interfície Web. Tanmateix, és fàcil detectar contingut en una aplicació (informació de contacte, publicitat, informació del perfil

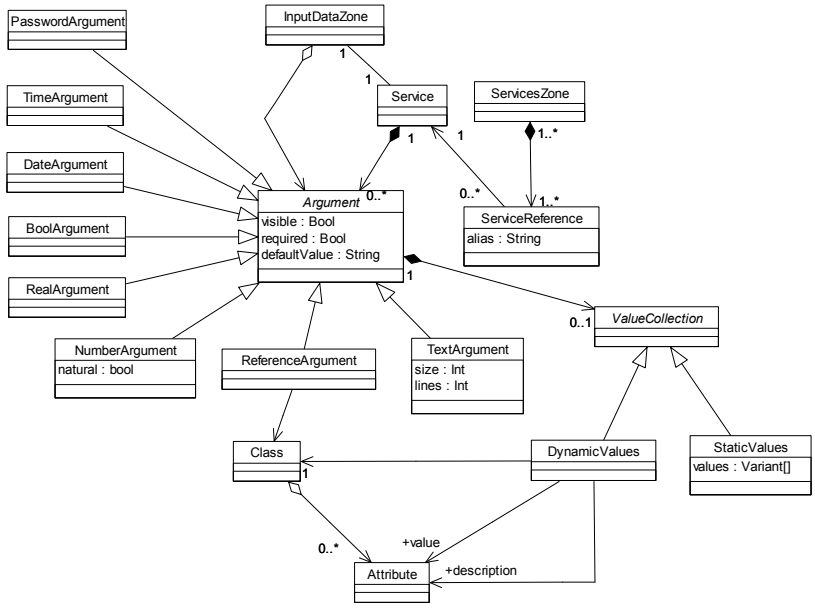


Figura A.5 – Model per a la Definició de Serveis

d'usuari) que no està contemplada en les zones vistes.

Per tal de donar suport a aquest tipus de necessitats, s'han creat les següents zones: *Page.AddUserInfoZone*, *Page.AddCustomZone* i *Page.AddCustomExternZone*.

A.2.9.1 Page.AddUserInfoZone

Afig la pàgina una zona de contingut que mostra la informació de l'usuari conectat (*login* i *rol*) i permet tancar la sessió.

Exemple:

```
$Page->AddUserInfoZone(‘‘userzone’’, ‘‘logged user’’);
```

A.2.9.2 Page.AddCustomZone

Afig a la pàgina una zona de continguts personalitzada de caràcter estàtic. Exemples de continguts que són especificats mitjançant aquesta zona són la d'informació institucional-empresa, els anuncis, missatges de benvinguda, etc.

Una zona personalitzada es defineix mitjançant un identificador i un àlies. Aquest identificador serà a la vegada el nom d'un fitxer que conté el contingut de la zona (ubicat al recurs *customZones*), en forma de HTML "lliure".

Exemple:

```
$Page->AddCustomZone(''ContactInformation'', ''About us'');
```

A.2.9.3 Page.AddCustomExternZone

Crea una novava zona de contingut a partir de la informació rebuda d'altra (pàgina d'una) aplicació Web externa. A partir d'una URL agafa el contingut HTML de la mateixa i l'inclou en la pàgina en una nova zona.

Aquest mecanisme permet reutilitzar desenvolupaments ja realitzats i residents inclús en altres servidors, com ara anuncis publicitaris, fragments de codi HTML, etc.

Exemple: El següent codi inserta en la pàgina una zona que mostra els deu millors "*tràilers*" del lloc web IMDb.

```
$Page->AddCustomExternZone(''trailerZone'', ''Best Trailers'',
''http://i.imdb.com/hotlink/top10_trailers_400.html'');
```

Apèndix B

Metamodel

En aquest annexe es mostra el Metamodel d'OOWS. Aquest metamodel té bàsicament dues funcions: (1) descriure d'una manera precisa les primitives de modelatge, i (2) servir com a base per definir l'eina CASE d'OOWS, incloent els editors de models i els transformadors.

A causa del tamany d'aquest metamodel, es mostrarà en paquets. Cadascun d'aquestos paquets respon a un tipus de requeriment concret. El metamodel ha sigut desenvolupat emprant Ecore [40], incloent les restriccions OCL que ha complir el un model per ser vàlid. La versió completa d'aquest metamodel pot obtenir-se de [128].

Les següents seccions mostren aquestos paquets en que s'ha dividit el model: *Diagrama d'Usuaris*, *Model de Navegació*, *Model de Presentació*, *Mecanismes d'Accés* i *Tipus de Dades*.

B.1 Diagrama d'Usuaris

Aquesta part del metamodel captura la informació relacionada amb els usuaris del sistema, representats per la classe *UserRol*. Un usuri té un identificador (nom de l'usuari) i un tipus, que pot ser *Anònim*, *Registrat* o *Sense Accés*.

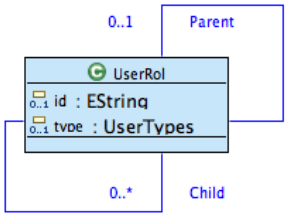


Figura B.1 – Metamodel: Diagrama d'Usuaris

Per capturar les relacions entre usuaris, un *UserRol* pot tenir un altre com a *Parent*, i pot especialitzar-se en diversos usuaris.

Restricció #1: Un usuari de tipus anonim no pot especialitzar-se d'un usuari enregistrat.

```
context UserRol
inv:
if self.type = UserTypes::Anonymous then
not self.parent.type = UserTypes::Registered
else true
endif
```

B.2 Model de Navegació

Aquesta part del metamodel captura les primitives principals d'OOWS: *Nodes Navegacionals*, *UAIs*, *Classes Navegacionals*, *Atributs Navegacionals*, *Operacions Navegacionals*, *Relacions Navegacionals*, etc.

A causa d'açò, aquesta porció del metamodel és la que més comportament captura. A continuació es mostren les restriccions en OCL que s'apliquen sobre els seus elements.

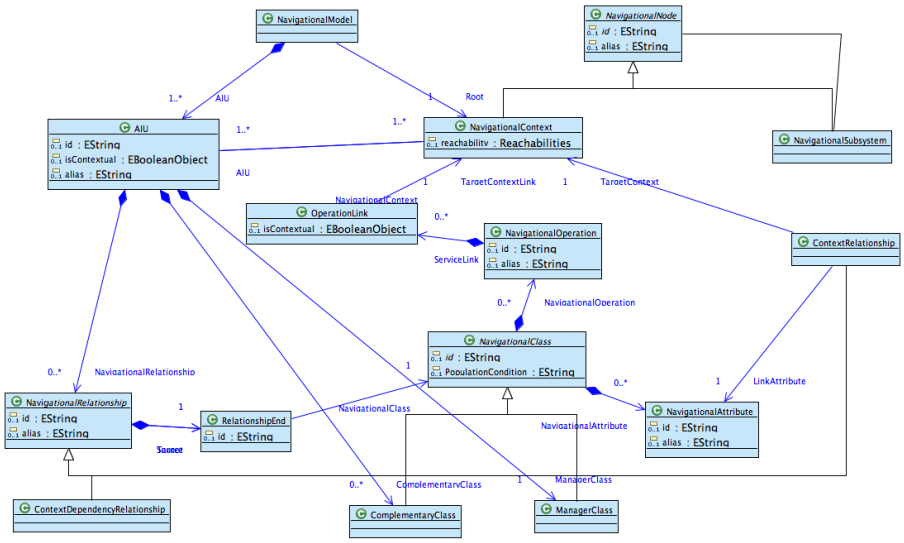


Figura B.2 – Metamodel: Model de Navegació

Restricció #2: Tot contexte de seqüència té almenys una UAI contextual.

context NavigationalContext
inv:
if self.reachability=Reachabilities::Sequence then
self.AIU->exists(e|e.isContextual = true)
else true endif

Restricció #3: Tots els atributs navegacionals de la classe navegacional han de ser atributs de la mateixa classe.

context NavigationalClass

```
inv:
self.NavigationalAttribute->forAll(att|
self.Class.Attribute->includes(att.Attribute))
```

Restricció #4: Totes les operacions navegacionals de la classe navegacional han de ser operacions de la mateixa classe.

```
context NavigationalClass
inv:
self.NavigationalOperation->forAll(op|
self.Class.Operation->includes(op.Operation))
```

Restricció #5: Les classes navegacionals de la relació navegacional són les mateixes que les classes de la relació estructural del diagrama de classes associada.

```
context NavigationalRelationship
inv:
self.Target <> self.Source and Association.allInstances()->one(a|
a.AssociationEnd->includesAll(Setself.Source.AssociationEnd,
self.Target.AssociationEnd))
```

Restricció #6: Si la relació de contexte té un atribut navegacional definit, la classe directora de totes les UAIs contextuals del node destí de la relació de contexte han de ser de la mateixa classe que la classe de la classe navegacional en la que està definit aquest atribut navegacional.

```
context ContextRelationship
```

```
inv:
if self.LinkAttribute.oclIsUndefined() then true else
let c:Class = Class.allInstances()->any(c|
c.Attribute->includes(self.LinkAttribute.Attribute)) in
if self.TargetContext.oclIsUndefined() then false else
self.TargetContext.AIU->forAll(a|a.ManagerClass.Class = c) endif
endif
```

Restricció #7: Si la relació de contexte no té un atribut navegacional definit, la classe directora de totes les UAIs contextuals del node destí de la relació de contexte han de ser classes directament relacionades amb la mateixa classe de la classe navegacional en la que està definida aquesta relació navegacional.

```
context ContextRelationship
inv:
Setself.Source.NavigationalClass,
self.Target.NavigationalClass->includesAll(self.TargetContext.AIU->select(a|
a.isContextual).ManagerClass)
```

Restricció #8: El node destí d'una relació de contexte ha de pertànyer al mateix mapa navegacional del node en el que està definida la relació de contexte.

```
context ContextRelationship
inv:
let source_aiu:AIU = AIU.allInstances()->any(a|
a.NavigationalRelationship->includes(self)) in
NavigationalContext.allInstances()->any(c|
```

c.AIU->includes(source_aiu)).navMap() = self.TargetContext.navMap()

Restricció #8 (Auxiliar): Les AIUs nomes pertanyen a un mapa navegacional.

context AIU inv: NavigationalContext.allInstances()->select(c| c.AIU->includes(self))->collect(x|x.navMap())->asSet()->size() =1

Restricció #8 (NavigationalNode:navMap()): Comprovació de ser fill d'un *UserRol*, si no, busca el seu Subsistems i s'invoca recursivament.

Context NavigationalNode:navMap()
post result= if UserRol.allInstances()->exists(u|
u.NavigationalNode->includes(self)) then
UserRol.allInstances()->any(u|
u.NavigationalNode->includes(self)) else
NavigationalSubsystem.allInstances()->any(s|
s.NavigationalNode->includes(self)).navMap() endif

B.3 Model de Presentació

Aquesta part del metamodel captura la informació relacionada amb els patrons de presentació (*Layout*), la ordenació que s'aplica als atributs navegacionals de les classes navegacionals (*OrderBy*) i la paginació (*Pagination*).

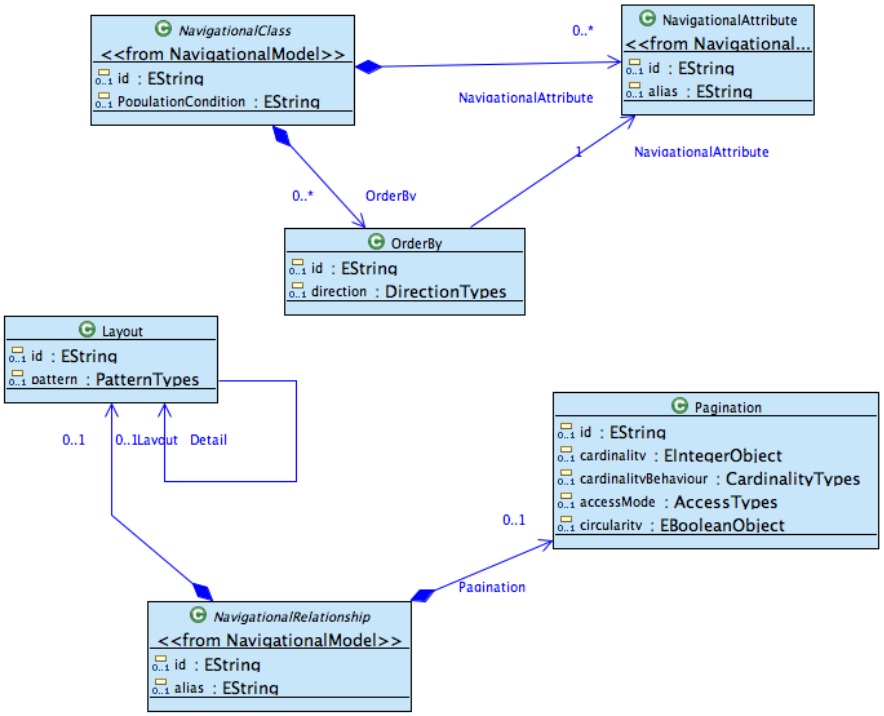


Figura B.3 – Metamodel: Model de Presentació

B.4 Mecanismes d'Accés

Aquesta part del metamodel captura la informació relacionada amb els mecanismes d'accés, *Index* i *Filter*, així com la vista de resultats (*SearchView*). Tota aquesta informació és capturada associada a la UAI que complementa (veure Capítol 6 la Secció 6.2).

B.5 Tipus de Dades

Per últim, aquesta part del metamodel conté els tipus de dades que s'empren per establir propietats que prenen valors controlats (enumerats).

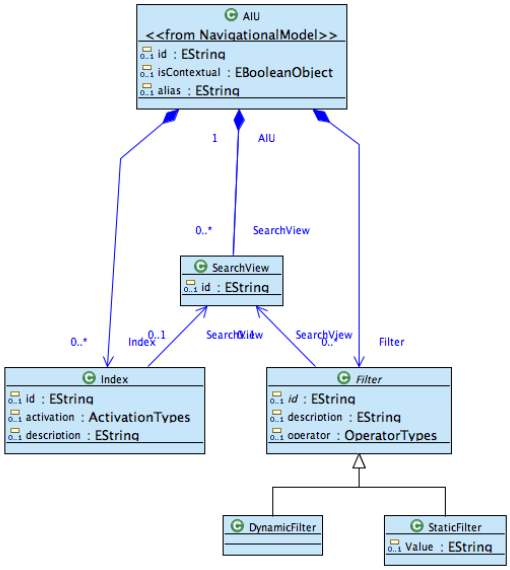


Figura B.4 – Metamodel: Mecanismes d'Accés

Així, apareixen els "enumerats": *DataTypes* (que conté els tipus de dades bàsics), *PatternTypes* (per indicar el tipus de *Layout*), *Reachabilities* (per indicar l'alcançabilitat dels nodes navegacionals), *UserTypes* (que indica els tipus d'usuaris), etc.

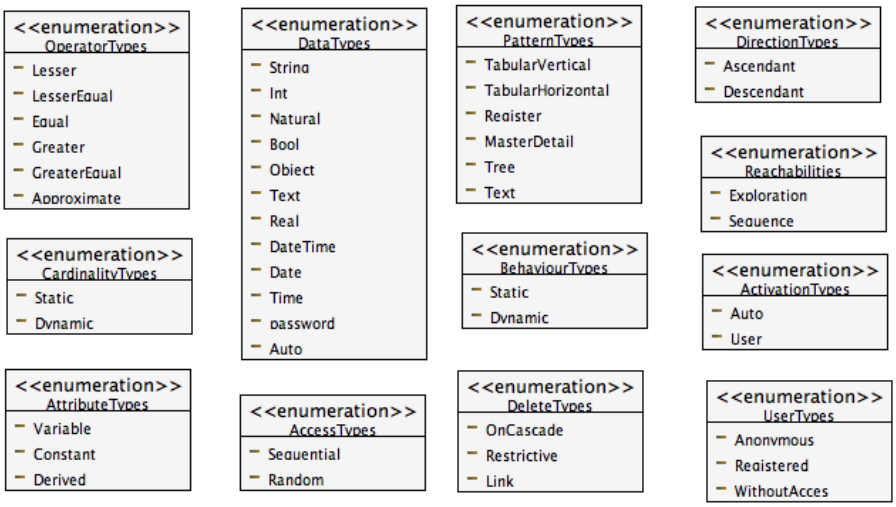


Figura B.5 – Metamodel: Tipus de Dades

Apèndix C

Cas d'Estudi: IMDb Lite

La *base de dades pel·lícules per Internet*, més coneguda com IMDb [58] és un repositori *online* d'informació relacionada amb el món del cine. Com diuen al lloc web oficial, és "the Earth's Biggest Movie Database" (la base de dades més gran del món sobre pel·lícules).

Aquest sistema emmagatzema qualsevol tipus d'informació relacionada amb les pel·lícules (*Movie*): ara detalls sobre la pel·lícula (producció, notes, duració, format, "*tràilers*", galeries de fotos, bandes sonores, els millors diàlegs, etc.); els participants en les pel·lícules (actors, directors, escriptors, productors, etc.); i informació sobre la posta en escena actualment (on s'estan projectant i planficacions temporals).

Qualsevol usuari pot registrar-se al sistema de manera que puguen introduir revisions crítiques sobre pel·lícules, o inclús valorar-se, establint un rànking de votacions. D'altra banda, els usuaris anònims sols poden emprar el sistema per explorar el catàleg, sense poder realitzar cap més operacions[1] (a menys que es registren).

El propòsit d'aquest annexe és descriure el model conceptual que represente de la manera més fidel el sistema real, realitzant una tasca

[1] Almenys fins a dia d'avuí, 21/12/2007

d'enginyeria inversa. Com que el sistema és suficientment gran, per tal de simplificar l'enteniment de tot el procés de modelatge i generació de codi, només es descriurà una part del sistema, la principal. Aquest subsistema, que anomenarem *IMDb Lite* és l'encarregat d'emmagatzemar la informació sobre les pel·lícules, els participants, les revisions dels usuaris i informació de planificació d'exposicions en cines.

Seguint l'aproximació OO-Method/OOWS presentada en aquest treball de tesi, el primer pas consisteix en realitzar una descripció conceptual del sistema. La Secció C.1 descriu els aspectes estructurals i de comportament del sistema mitjançant OO-Method, mentre que la Secció C.2 descriu l'extensió conceptual web emprant la proposta OOWS.

Finalment, la Secció C.3 presenta el resultat de la compilació d'aquestos models i el codi obtés amb el Framework d'Intertícies Web (Annexe A). Aquest cas d'estudi està disponible online de manera completa en [128].

C.1 Model Conceptual amb OO-Method

El primer pas consisteix en descriure els aspectes d'estructura d'informació i funcionals. Aquestos requeriments són capturats en OO-Method per mig d'un *Diagrama de Classes*, un *Model Dinàmic* i un *Model Funcional*.

Les següents seccions mostren la descripció d'aquestos models aplicats a *IMDb Lite*.

C.1.1 Diagrama de Classes

El *Diagrama de Classes* captura els aspectes estructurals del sistema des d'un punt de vista orientat a objectes: classes, atributs, operacions i relacions entre classes.

D'acord al sistema *IMDb Lite*, els objectes d'interés són les pel·lícules, els participants, exhibicions, crítiques dels usuaris, etc. La FiguraC.1

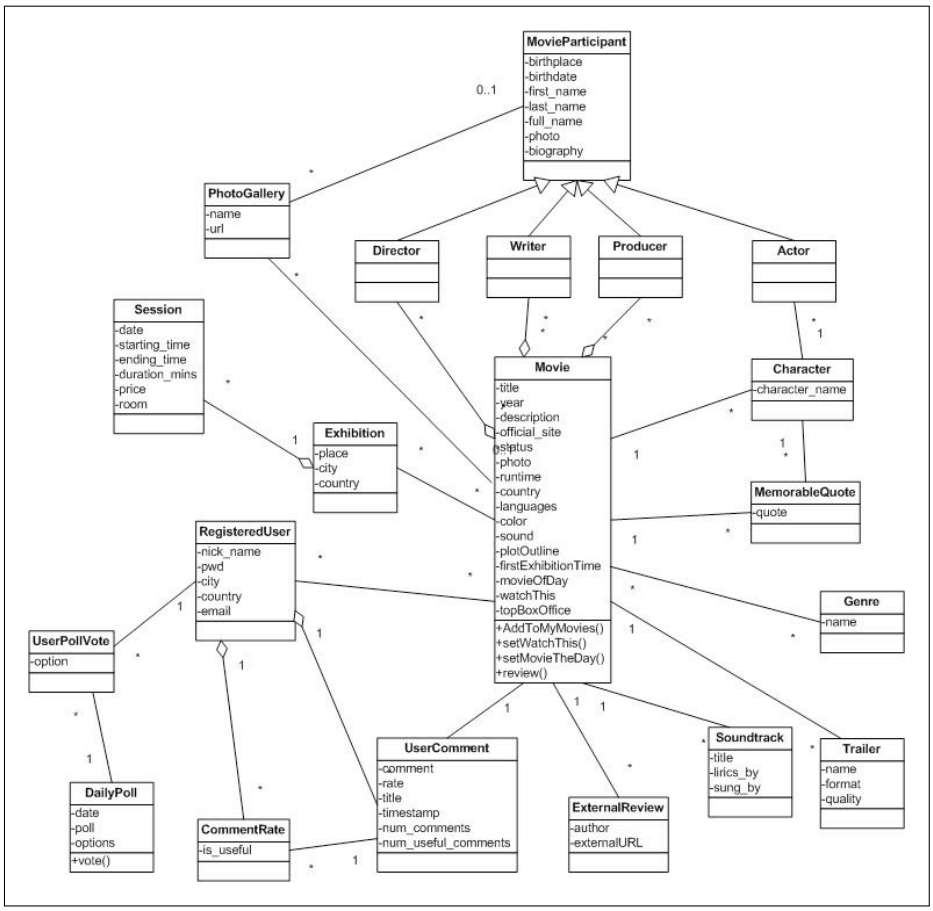


Figura C.1 – *IMDb Lite*: Model Estructural

mostra el diagrama de classes construït per al sistema i que s'agafarà com a base en el desenvolupament.

Com es pot apreciar, aquesta figura emfatisa la part del sistema que està relacionada amb la informació de les pel·lícules: la classe *Movie* és la principal del diagrama. Aquesta classe introdueix les propietats d'una pel·lícula: el seu títol, l'any de producció, una descripció general, la URL oficial del lloc web de la pel·lícula, l'estat de producció ("filmació",

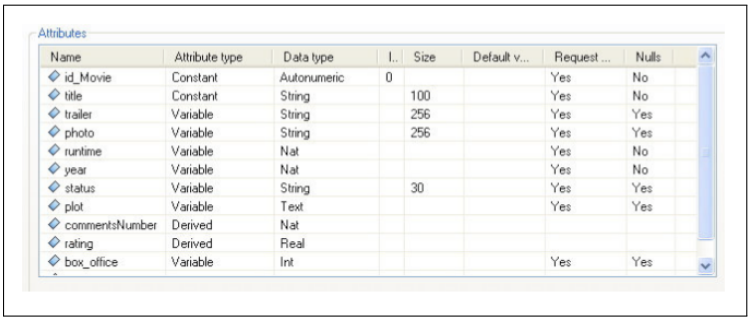


Figura C.2 – *IMDb Lite*: Atributs de la Classe *Movie*

"post-producció", "pre-estreno", etc.), una foto principal, els idiomes en que està disponible, el sistema de so, etc.

A més a més, una pel·lícula pot ser marcada amb diferents etiquetes per indicar que és la "pel·lícula del dia" o "pel·lícula recomanada" ("*watch This*"), que després s'emprarà en l'aplicació Web per mostrar suggeriments.

Emprant la propietat "firstTimeExhibition" (data de la primera exhibició de la pel·lícula), es pot derivar dinàmicament les pel·lícules que estan a punt d'estrenar-se ("*coming soon*"). Açò es representa amb OO-Method com un atribut derivat. *OlivaNOVA* disposa d'un llenguatge de definció de fòrmules complexes, mitjançant operadors lògics, de col·leccions i aritmètics, que proporciona un ample espectre de possibilitats.

Per al cas del càlcul de l'atribut derivat *coming_soon*, aquest es crea associat a la classe *Movie*, de tipus booleà, i la seua fòrmula de derivació podria representar-se com (si considerem 15 dies com el marge per considerar que s'estrena prompte):

```
Movie.coming_soon = ( date_diff(Movie.firstExhibitionTime,toda(),"d")
< 15 and date_diff(Movie.firstExhibitionTime,toda(),"d") > 0
```

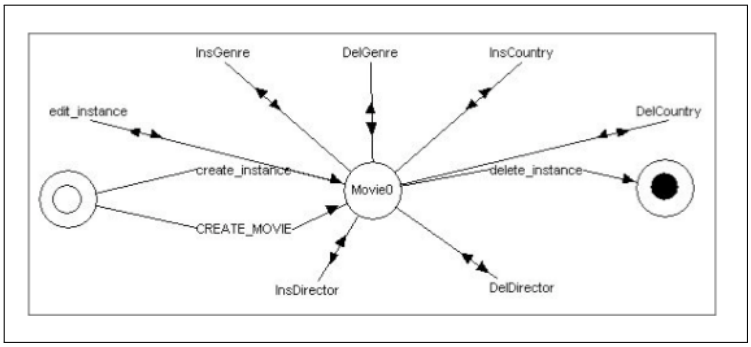


Figura C.3 – *IMDb Lite*: DTE per a la classe *Movie*

És a dir, una pel·lícula està a punt d'estrenar-se si hi ha menys de 15 dies entre avuí i la data d'estrena, i la data d'estrena no ha passat.

C.1.2 El Model Dinàmic

A més de l'estructura de classes del sistema, hem de considerar certs aspectes de comportament. El *Model Dinàmic* permet descriure els cicles de vida vàlids dels objectes, així com certa comunicació interobjectual.

Per a realitzar aquestes especificacions, OO-Method empra dos tipus de diagrames: el *Diagrama de Transició entre Estats*, i el *Diagrama d'Interacció*.

El *Diagrama de Transició entre Estats* o DTE, s'empra per descriure el comportament d'un objecte descrivint el seu cicle de vida. Per a descriure aquest cicle de vida, cal desciure una ordenació en la manera que es poden executar les seues operacions. Tota classe ha de tenir un DTE associat.

La Figura C.3 mostra el cicle de vida associat a la classe pel·lícula ("Movie").

D'altra banda, amb el *Diagrama d'Interacció* podem especificar la comunicació entre objectes. Tanmateix, no ha calgut definir res d'aquest model per al sistema *IMDb Lite*.

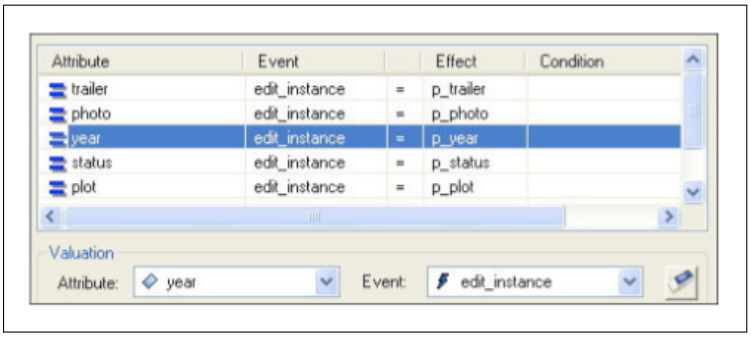


Figura C.4 – *IMDb Lite*: Model Funcional de la classe *Movie*

C.1.3 El Model Funcional

Mitjançant el *Model Funcional* es captura la semàntica associada al canvi d'estat dels objectes com conseqüència de l'execució dels seus events. Emprant un formlisme subjacent (basat en la lògica dinàmica [53]), per a cada atribut variable es defineix una fòrmula que defineix com un o diferentes events poden modificar aquest atribut.

La Figura C.4 mostra un fragment del Model Funcional apreciant-se l'efecte de l'event *edit_ instance* sobre la classe pel·lícula (*Movie*).

C.2 Extensions Web amb OOWS

Una vegada descrits els requeriments d'estructura d'informació i funcionals, el següent pas és descriure els requeriments navegacionals del sistema. Seguint l'aproximació OOWS, cal definir els següents diagrames: (1) un *diagrama d'usuaris*; (2) un *mapa navegacional* per a cada usuari del diagrama d'usuaris, descrivint els seus nodes; i (3) els requeriments de presentació associats a tots els nodes navegacionals detectats.

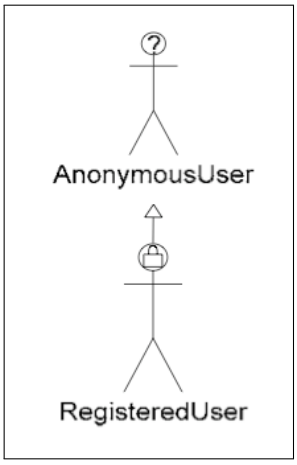


Figura C.5 – *IMDb Lite*: Diagrama d'Usuaris

C.2.1 Diagrama d'Usuaris

En *IMDb Lite* hi ha dos tipus d'usuaris: els *AnonymousUser* (Usuaris Anònims) i els *RegisteredUser* (Usuaris Registrats). Ambdós tipus poden explorar la base de dades de pel·lícules, però sols els usuaris registrats poden introduir les seues opinions i vots (crítiques).

La Figura C.5 mostra aquest Diagrama d'Usuaris. Els *AnonymousUser* estan etiquetats amb la marca "?", indicant que no necessiten identificar-se per accedir al sistema. Els *RegisteredUser* estan especialitzats dels usuaris anònims, indicant que heretaran les propietats navegacionals d'aquestos. Seguint l'aproximació OO-Method (que introdueix els usuaris com part de l'estructura d'informació del sistema), aquestos usuaris formen part del diagrama de classes de l'aplicació.

C.2.2 Model Navegacional

Al següent pas s'ha de definir un *Mapa Navegacional* per a cada tipus d'usuari. Aquest Mapa Navegacional defineix l'accessibilitat de l'usuari

al sistema. La Figura C.6 a la pàgina següent mostra el mapa navegacional per als usuaris registrats. Aquest mapa està compost per 18 contextes navegacionals i un subsistema navegacional.

Cadascun dels contextes navegacionals definits proporciona un vista diferent del diagrama de classes, proporcionant a l'usuari diferents punts de vista sobre la mateixa estructura d'informació. El contexte "*Now Playing*" proporciona informació sobre les pel·lícules que estan exhibint-se actualment, mentre que el contexte "*Showtime and Tickets*" mostra els llocs (cines) on s'exhibeixen.

El contexte "*My Movies*" permet als usuaris marcar les seues pel·lícules preferides (donar el seu vot o crítica).

De tots aquestos contextes, hi ha 14 que estan marcats com d'exploració (etiquetats amb una "E"). Açò significa que aquestos nodes sempre estaran disponibles per accedir, estiguem on estiguem dins el sistema. La resta de nodes són de seqüència (etiquetats amb una "S"). Açò significa que sols es podrà accedir a aquestos nodes a través d'un camí navegacional predefinit.

El contexte anomenat "*Home*", d'exploració, té a l'enllaç d'exploració (fletxa discontínua) associada l'etiqueta "Home". Açò significa que aquest contexte serà el node per defecte quan un usuari registrat es conecte al sistema. Aquest contexte proporciona informació sobre: les pel·lícules actuals ("*Trailers and more*"), la votació diària ("*daily poll*"), les pel·lícules del dia ("*movie of the day*"), etc. Tots aquestos continguts són definits mitjançant *Unitats d'Abstracció d'Informació* (UAI), que composen el contexte. Com es pot apreciar, a pesar que aquest node captura gran quantitat d'infomació, aquesta està ben organitzada i estructurada amb aquestes UAIs. La Figura C.7 a la pàgina 352 mostra la descripció d'aquest contexte "Home".

La UAI "*Whatch This: Trailers and More*" és l'encarregada de proporcionar als usuaris informació sobre els títols de les pel·lícules que estan

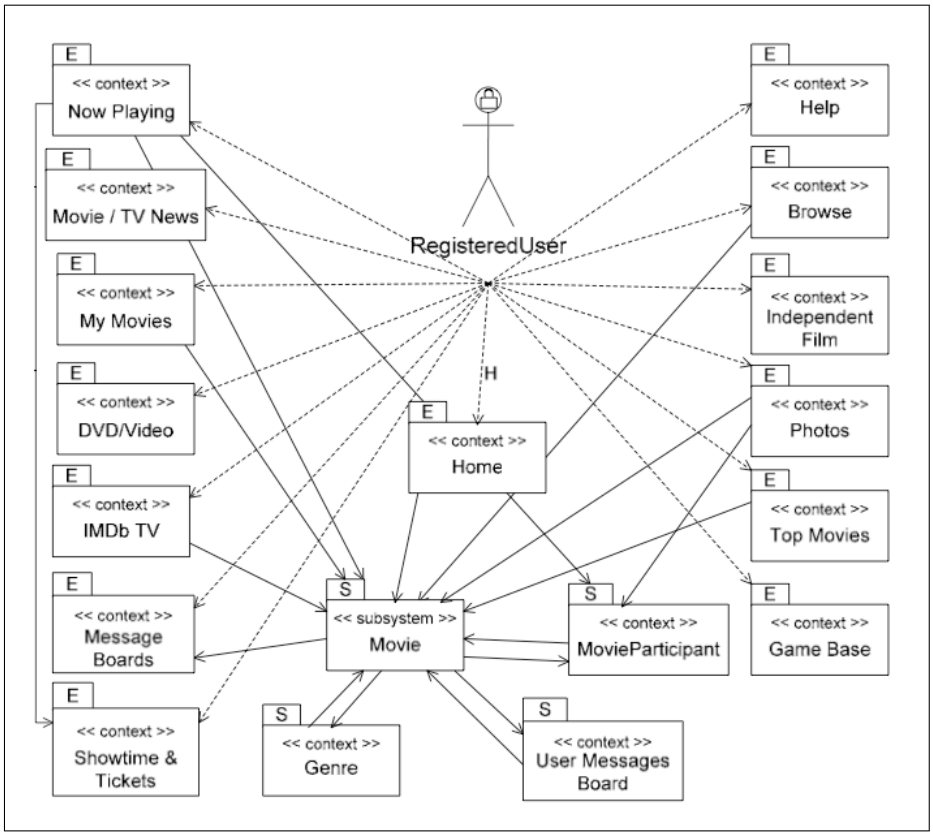


Figura C.6 – *IMDb Lite*: Mapa Navegacional per al *RegisteredUser*

marcades amb la propietat *watchThis*. Aquest requeriment s'ha definit especificant la classe *Movie*, amb l'atribut navegacional *title* i una condició de filtrat de població: $watchThis = TRUE$. Aquesta UAI té definides dues capacitats de navegació, mitjançant dos relacions de contexte. Una que duu al contexte *Trailer*, que està dins el subsistema *Promotional* (que a la vegada està dins el subsistema *Movie*); i l'altra opció de navegació ens duu al contexte *Main Details*, dins el subsistema *Overview* (també dins el subsistema *Movie*). La Figura C.8 a la pàgina 354 mostra aquesta

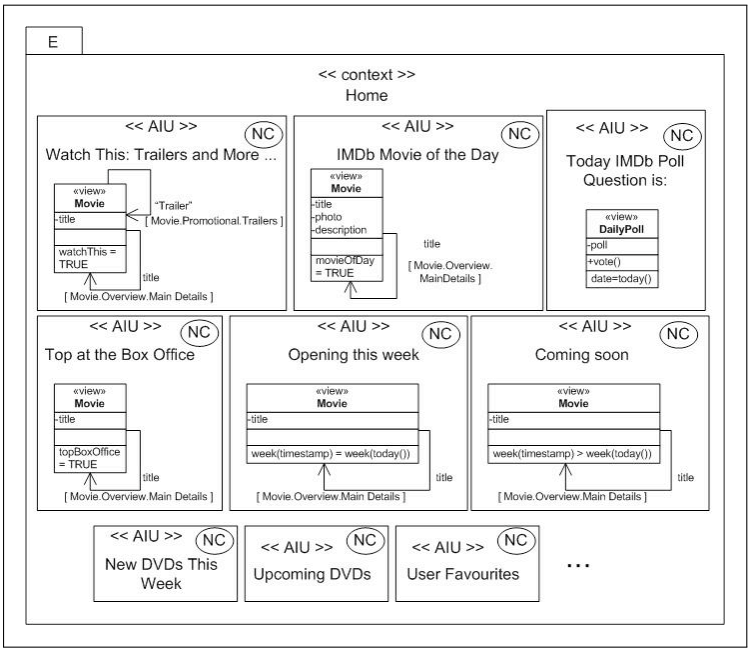


Figura C.7 – *IMDb Lite*: Contexte *Home*

estructura del sistema en subsistemes.

Encara al contexte *Home*, la UAI "*Today IMDb Poll Question is:*" proporciona a l'usuari informació sobre la votació diària ("*DailyPoll*"). Per representar el requeriment que és "votació d'avuí", s'ha especificat una condició de filtrat de població a la classe *DialyPoll* indicant que la seua data es la d'avuí: *date = today()*[2]. Aquesta UAI també permet a un usuari registrat realitzar una votació, ja que apareix el mètode per a votar ("vote").

De la mateixa manera, les altres UAIs proporcionen a l'usuari informació addicional sobre altres aspectes del sistema (pel·lícula del dia, pròximes estrenes, etc.). Totes aquestes UAIs estan marcades com *No Contextuals* (etiqueta NC) perquè no necessiten informació contextual

[2]Today() és una operació de l'entorn que ens permet recuperar la data del sistema.

(que provinga com a resultat d'haver activat una navegació) per recuperar la informació que defineixen. La Figura C.14 a la pàgina 360 mostra aquesta pàgina implementada.

A causa de la gran quantitat d'informació disponible per a una pel·lícula, l'estructura navegacional s'ha organitzat per subsistemes, de manera que faciliten la navegació pel sistema. Així, al mapa navegacional, s'ha creat el subsistema "*Movie*". Aquest subsistema serà doncs responsable de proporcionar tota la informació del catàleg sobre una pel·lícula, organitzat a la seua vegada per diferents subsistemes: "*Overview*" (que mostra informació resumida de les pel·lícules), "*Awards and Reviews*" (que es centra en mostrar informació sobre els premis que s'han otorgat), "*Fun Stuff*" (informació curiosa i divertida al voltant d'una pel·lícula), etc. Dins aquest subsistema "*Movie*", el subsistema "*Overview*" està etiquetat amb una "H", fet que el converteix en el node per defecte.

La Figura C.8 a la pàgina següent mostra aquesta organització del subsistema "*Movie*".

Les Figures C.9 i C.10 mostren una vista expandida dels subsistemes "*Promotional*" i "*Overview*", respectivament.

El subsistema "*Movie.Promotional*" permet als usuaris navegar a través de la (*Tagline* (trama), els *Trailers*, els*Posters* i la *Photo Gallery* d'una pel·lícula.

El subsistema *Movie.Overview* proporciona la principal informació sobre una pel·lícula (es podria considerar com la part més important de la base de dades de IMDb): *Main Details* (detalls principals), *Full Cast and Crew* (la gent que participa en la pel·lícula), *Combined Details* (paregut als detalls principals, però mostrat d'una altra manera) i *Company Credits* (informació sobre la companyia).

Dins el subsistema *Movie.Overview* es troba el contexte navegacional *Main Details*, que és l'encarregat de proporcionar la informació bàsica

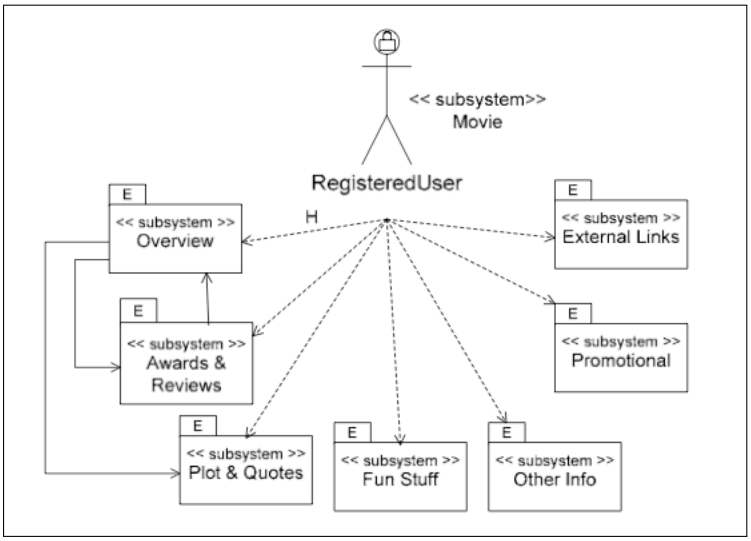


Figura C.8 – *IMDb Lite*: Subsistema *Movie*

d'una pel·lícula: títol, resum, duració, país, idiomes, etc. També proporciona informació sobre els *Directors*, *Writers* (guionistes) i *Producers* (productors), a més de les frases cèl·lebres (*memorable quotes*) dels actors en la pel·lícula (Figura C.11 a la pàgina 357).

Aquest contexte *Main Details* està composat per 4 UAIs contextuals (marcades amb una C) que mostraran informació relacionada amb la pel·lícula seleccionada. També hi ha una UAI no contextual (*Message Board*), on els usuaris registrats deixen els seus comentaris sobre les pel·lícules.

Dins aquest contexte, es pot considerar que la UAI *Main Details* és la principal, ja que és la que mostra la informació de la pel·lícula (títol, any, resum, ...). També permet (1) afegir marcar (amb el mètode *addToMyMovies()* una pel·lícula com preferida (apareixerà al contexte *MyMovies*; (2) valorar una pel·lícula (amb el mètode *rate()*); i (3) escriure una revisió de la pel·lícula (amb el mètode *review*).

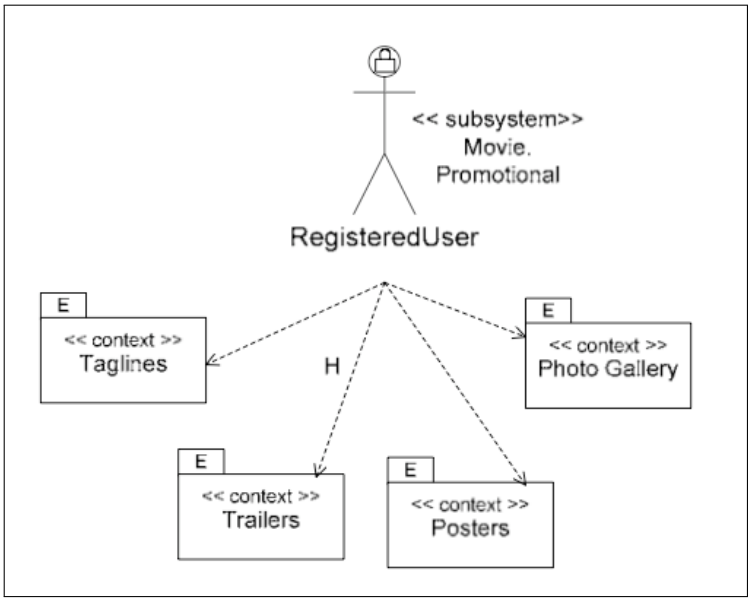


Figura C.9 – *IMDb Lite*: Subsistema *Movie.Promotional*

C.2.3 Model de Presentació

Després d'haver especificat el model navegacional, es deuen especificar els requeriments de presentació emprant el *Model de Presentació*.

IMDb segueix un princpi molt senzill i homogeni de mostrar la informació: per cada entitat (instància d'una classe), tota la informació es mostra a partir d'aquesta en mode registre. És a dir, totes les relacions "un-a-un" empren el patró *Registre*, i totes les relacions "un-a-molts" empren el patró *Mestre-Detall*, amb el detall establert al patró *Registre*.

Aquest fet condueix a un Model de Presentació prou senzill. La Figura C.12 a la pàgina 358 mostra un exemple representatiu de l'especificació d'aquestos requeriments de presentació aplicats al contexte "*Movie.Overview.MainDetails*".

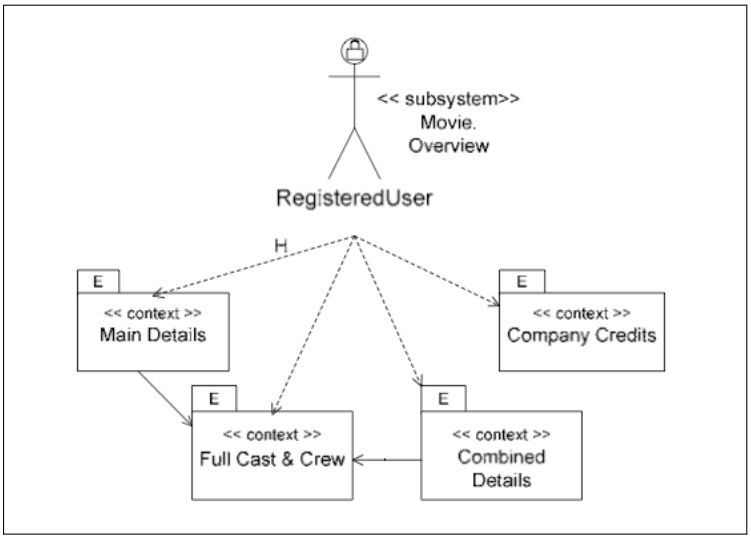


Figura C.10 – *IMDb Lite*: Subsistema *Movie.Overview*

C.3 Implementació de l'aplicació Web

Aquesta secció descriu com obtenir un prototipus de l'aplicació *IMDb Lite*. Després d'haver construït l'especificació conceptual del sistema, arribem a la fase d'implementació. El següent pas consisteix en aplicar les transformacions dels models web. Per fer-ho, ha sigut necessari exportar els models realitzats amb *OlivaNOVA* a XML, importar-los a l'eina *OOWS Visual Editor* per a finalment poder aplicar aquestes transformacions. El Capítol 9 explica els detalls de com assolir-ho. Al final, s'aconsegueix una aplicació web implementada amb el Framework d'Interfícies Web definit a l'Annexe A.

La Figura C.13 a la pàgina 359 mostra el fitxer de configuració del framework generat per a l'aplicació. Si ens fixem, el compilador ha asignat l'estil per defecte anomentat IMDb (el nom de l'aplicació). Atenent a l'estratègia d'introducció d'aspectes de disseny gràfic, s'ha

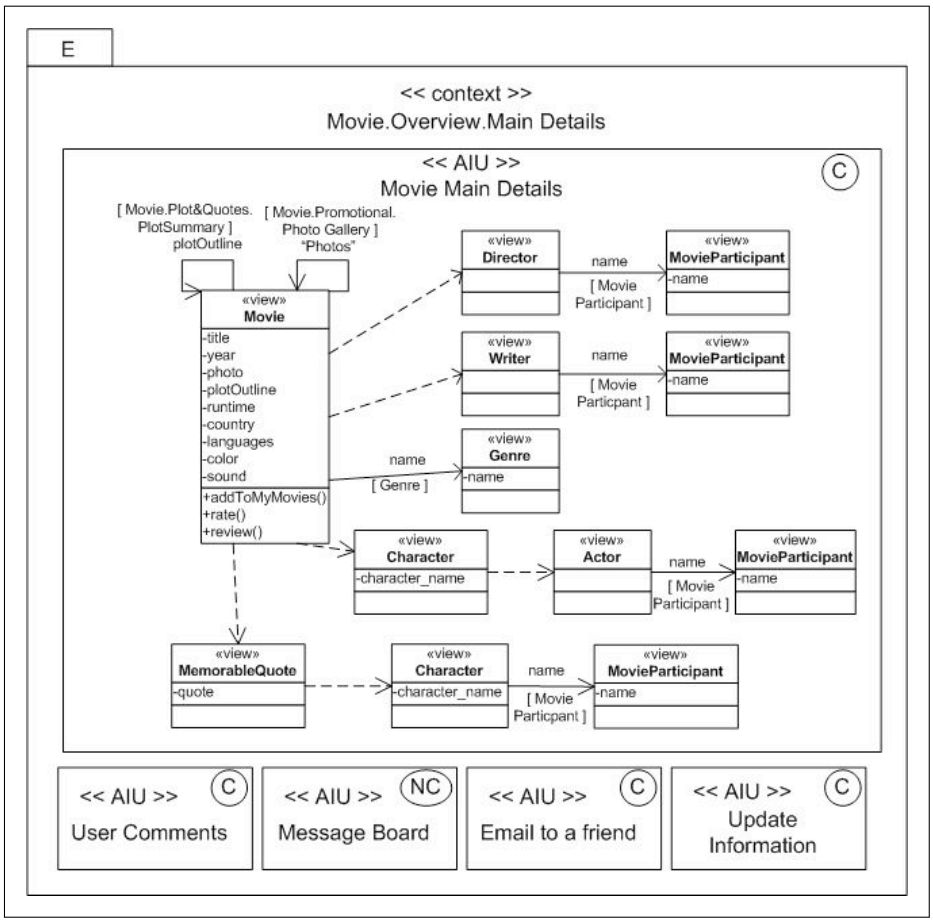


Figura C.11 – *IMDb Lite*: Contexte *Movie.Overview.MainDetails*

construït aquest estil de manera que tinga una semblança amb l'aplicació Web real IMDb.

Aquest fitxer de configuració defineix dos rols diferents (*AddRol*) que provenen del Diagrama d'Usuaris especificat. Tal com apareix en aquest diagrama, l'usuari *RegisteredUser* és una especialització de

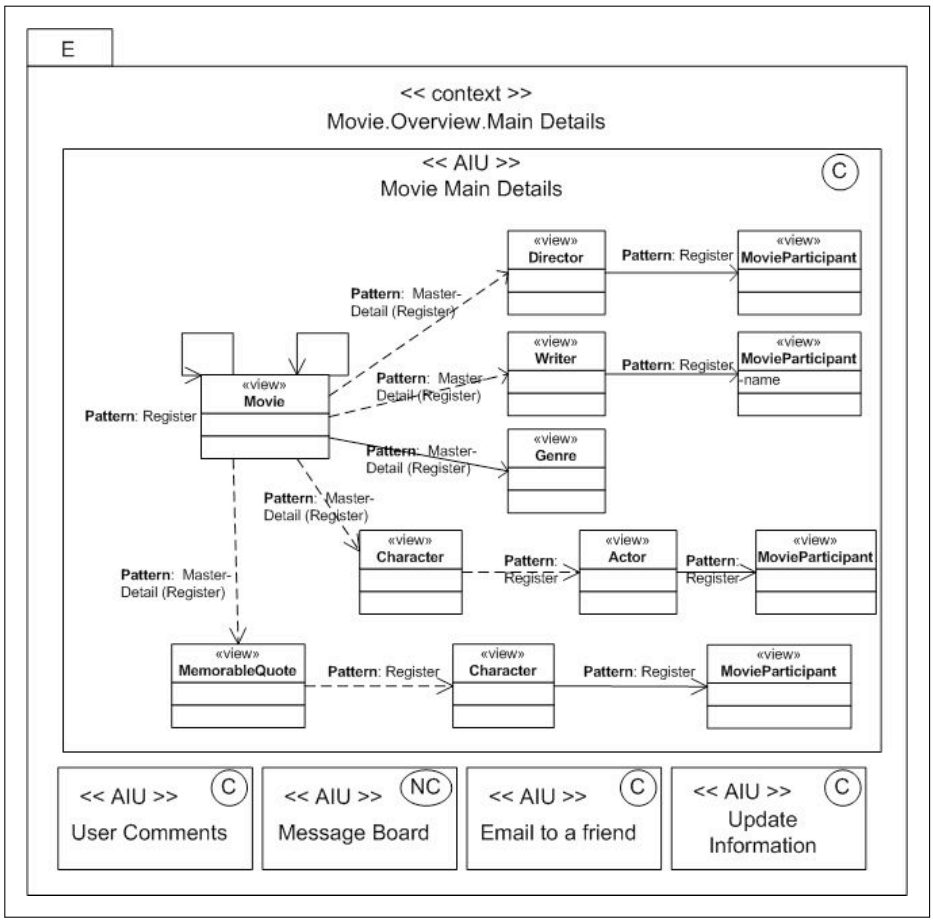


Figura C.12 – *IMDb Lite*: Contexte de Presentació *Movie.Overview.MainDetails*

l'usuari *AnonymousUser*, i així apareix a la seua sentència de creació[3].

Per cada contexte navegacional especificat s'ha creat una pàgina web (*AddPage*). Per cada subsistema navegacional s'ha creat un grup de pàgines (*AddPageGroup*), incloent-li tots els nodes que conté. La pàgina per defecte (*SetHomePage*) ha sigut també marcada (*Home*).

[3]Veure l'últim argument de l'operació AddRol associada al *RegisteredUser*.

```
<?php
include_once "../Framework/ApplicationBegin.php";

     $Application= new Application("IMDB","IMDB");

     $Application->AddRol("AnonymousUser","","","");
     $Application->
               AddRol("RegisteredUser","RegisteredUser","RegisteredUser_MVAgentValidation","AnonymousUser");

     $Application->AddPage("NowPlaying","NOW PLAYING","","AnonymousUser","always");
     $Application->AddPage("MovieTVNews","MOVIE/TV NEWS","","AnonymousUser","always");
     $Application->AddPage("MyMovies","MY MOVIES","","AnonymousUser","always");
     $Application->AddPage("DVD/Video","DVD/VIDEO","","AnonymousUser","always");
     $Application->AddPage("IMDbTV","IMDb TV","","AnonymousUser","always");
     $Application->AddPage("MessageBoards","MESSAGE BOARDS","","AnonymousUser","always");
     $Application->AddPage("Showtime&Tickets","SHOWTIME&TICKETS","","AnonymousUser","always");
     $Application->AddPage("GameBase","GAME BASE","","AnonymousUser","always");

     $Application->AddPage("Home","Home","","AnonymousUser","always");
     ...

     $Application->AddPageGroup("PG_BrowseMovie","Movie","","AnonymousUser","always",);

     $Application->AddPageGroup("PG_Overview","Overview","Movie","AnonymousUser","always",);
     $Application->AddPage("MainDetails","main details","PG_Overview","AnonymousUser","always");
     $Application->AddPage("CombinedDetails","combined details","PG_Overview","AnonymousUser","always");
     $Application->AddPage("FullCastAndCrew","full cast and crew","PG_Overview","AnonymousUser","always");
     $Application->AddPage("CompanyCredits","company credits","PG_Overview","AnonymousUser","always");

     $Application->AddPageGroup("PG_Awards&Reviews","Awards & Reviews","Movie","AnonymousUser","always",);
     $Application->AddPage("UserComments","user comments","PG_Awards&Reviews","AnonymousUser","always");
     ...

     $Application->SetDefaultStyle("IMDB")

     $Application->SetHomePage("Home","AnonymousUser");

include_once "../Framework/ApplicationEnd.php";
?>
```

Figura C.13 – *IMDb Lite*: Fitxer de configuració d'aplicació generat

La Figura C.14 a la pàgina següent mostra la pàgina *Home* recupera informació variada, segons descriu el seu contexte navegacional a la Figura C.7 a la pàgina 352: *Watch This*, *Today's Poll*, *IMDb Movie of the Day*, etc. La part de la pàgina web que mostra la informació que especifica la UAI *Watch This* proporciona a l'usuari el títol de les pel·lícules marcades com recomanades. Aquest títol serveix com àncora per navegar al contexte *Movie.Overview.MainDetails*, tal i com s'ha especificat a la definició d'aquest contexte.

La UAI *IMDb Movie of the Day* mostra el títol, foto i descripció de la pel·lícula establerta com a "del dia" (a través de la propietat *movieOfDay*). La foto s'ha definit com àncora per navegar con contexte *Movie.Overview.MainDetails*.

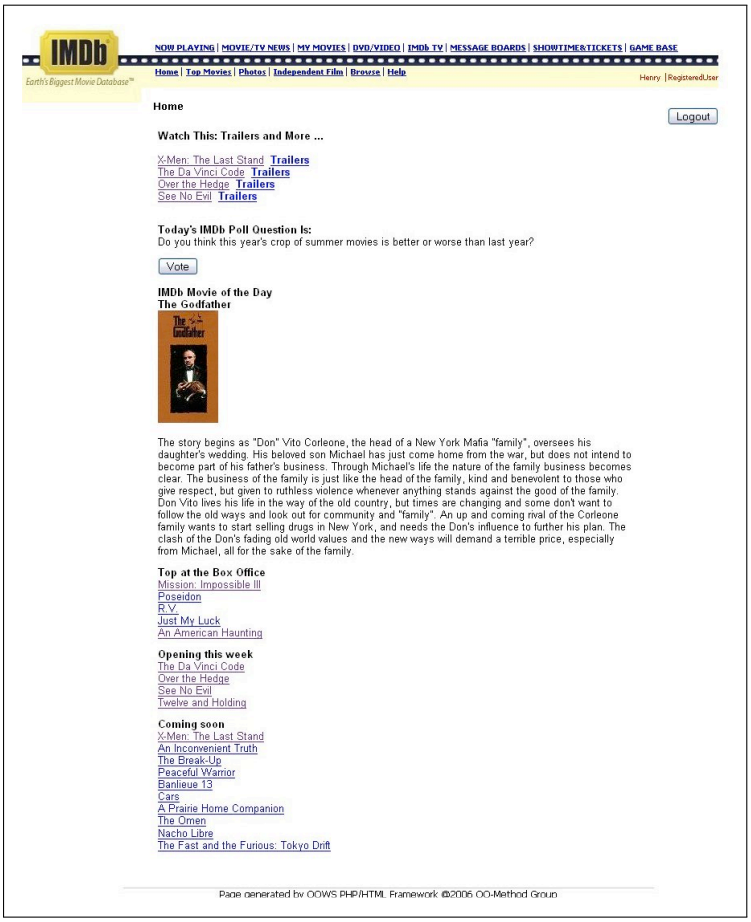


Figura C.14 – *IMDb Lite*: Implementació de la Pàgina *Home*

La pàgina *MainDetails* prové del contexte *MainDetails* que està dins el subsistema *Movie.Overview*. Aquest requeriment d'inclusió pot veure's al fitxer de configuració del sistema. El codi d'aquesta pàgina web es mostra a la Figura C.15 a la pàgina següent.

Aquesta pàgina web recupera tota la informació bàsica d'una pel·lícula: títol, any, foto, resum ("*PlotOutline*"). Per cadas-

```
<?php
include_once "../../Framework/PageBegin.php";

     $Page=new Page("Movie Overview Main Details","RegisteredUser");
     $Page->AddUserInfoZone();

     $InfoZone = $Page->AddInformationZone("AIU-1","Movie Main Details.","Movie");
     $InfoZone->AddField ("title","Title");
     $InfoZone->AddField("year","Year");
     $InfoZone->AddImageField ("photo","Photo");
     $InfoZone->AddField("plotOutline","Plot");
     $InfoZone->AddField("country","Country");
     $InfoZone->AddField("languages","Languages");
     $InfoZone->AddField("runtime","Runtime");
     $InfoZone->AddField("color","Color");
     $InfoZone->AddField("sound","Sound");
     $InfoZone->plotOutline->AddInternLinkTo("Movie_Plot&Quotes_PlotSummary");
     $InfoZone->photo->AddInternLinkTo("Movie_Promotions_PhotoGallery");

     $ServiceZone = $InfoZone->AddServicesZone("movieServices","");
     $ServiceZone->AddServiceReference("Movie_AddToMyMovies","Add to my Movies");
     $ServiceZone->AddServiceReference("Movie_rate","Rate it");
     $ServiceZone->AddServiceReference("Movie_review","Add a Review");

     $InfoZone->AddDetailRelationship("Director");
     $InfoZone->RelatedDirector->AddRelatedField("name","Name","MovieParticipant");
     $InfoZone->RelatedDirector->MovieParticipant_name->AddInternLinkTo("Movie_Participant");

     $InfoZone->AddDetailRelationship("Writer");
     $InfoZone->RelatedWriter->AddRelatedField("name","Name","MovieParticipant");
     $InfoZone->RelatedWriter->MovieParticipant_writer->AddInternLinkTo("Movie_Participant");

     $InfoZone->AddDetailRelationship("Genre");
     $InfoZone->RelatedGenre->AddField("name","");
     $InfoZone->RelatedGenre->name->AddInternLinkTo("Genres");

     $InfoZone->AddDetailRelationship("Character");
     $InfoZone->RelatedActorParticipant->AddField("character_name","Character");
     $InfoZone->RelatedActorParticipant->AddRelatedField("name","Name","Actor.Character");
     $InfoZone->RelatedActorParticipant->Actor_Character_name->AddInternLinkTo("Movie_Participant");

     $InfoZone->AddDetailRelationship("MemorableQuote");
     $InfoZone->RelatedMemorableQuote->AddField("quote","Quote");
     $InfoZone->RelatedMemorableQuote->AddRelatedField("character_name","Character","Character");
     $InfoZone->RelatedMemorableQuote->AddRelatedField("name","Name","Character.MovieParticipant");
     $InfoZone->RelatedMemorableQuote->Character_MovieParticipant_name->AddInternLinkTo("Movie_Participant");

        $InfoZone = $Page->AddInformationZone("AIU-2","User Comments.","User");
        ...

include_once "../../Framework/PageEnd.php";
?>
```

Figura C.15 – *IMDb Lite*: Codi Pàgina Web *Movie.Overview.MainDetails*

cun d'aquestos camps a visualitzar, pot veure's al codi del framework generat com apareix una operació *AddField* associada a la UAI on està definit (en aquest cas, a la *AIU-1*). L'atribut *plotOutline* fou especificat com àncora per navegar a la pàgina "*Movie.PlotandQuotes.PlotSummary*". Açò està representat al codi mitjançant la sentència: *plotOutline→AddInternalLinkTo*.

L'aparença d'aquesta pàgina *Movie.Overview.MainDetails* pot veure's a la Figura C.16 a la pàgina següent.

Com es pot apreciar, l'estil de visualització és semblant al que empra IMDb. Això es deu a que les regles de visualització emprades s'han dissenyat per recrear l'aparença d'aquesta aplicació.

Per demostrar la reutilització d'aquestes regles de visualització, s'-

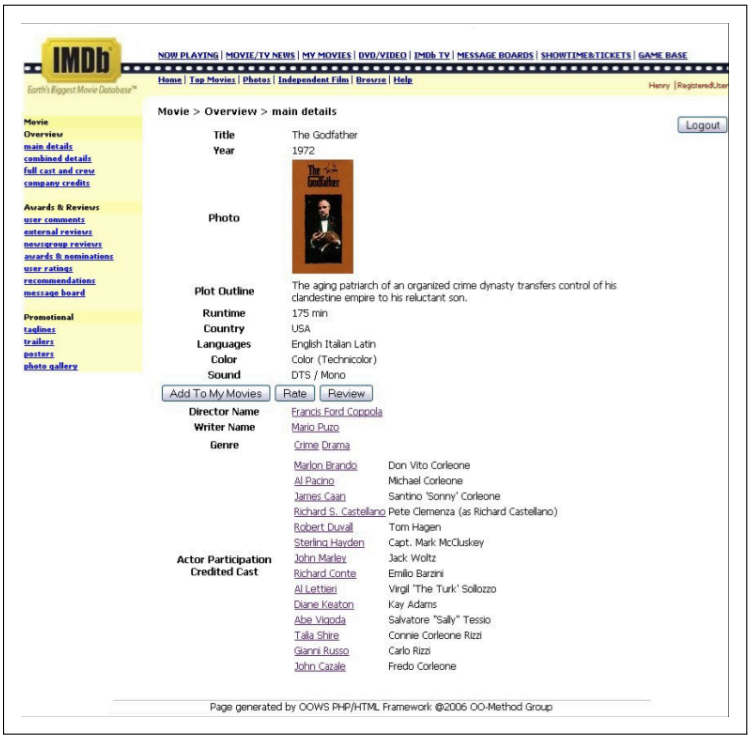


Figura C.16 – *IMDb Lite*: Pàgina Web *Movie.Overview.MainDetails (estil IMDb)*

ha decidit emprar en aquesta aplicació web un altre estil prèviament definit i utilitzat en altres desenolupaments. Aquest nou estil implementa l'estil visual que empra l'Universitat Politècnica de València (http://www.upv.es) a dia d'avuí.

Per aconseguir-ho, només cal anar al fitxer configuració de l'aplicació, i canviar l'argument de l'operació *SetDefaultStyle* per:

```
$Application->SetDefaultStyle("UPV-like");
```

La Figura C.17 a la pàgina següent mostra l'aplicació Web *IMDb Lite*

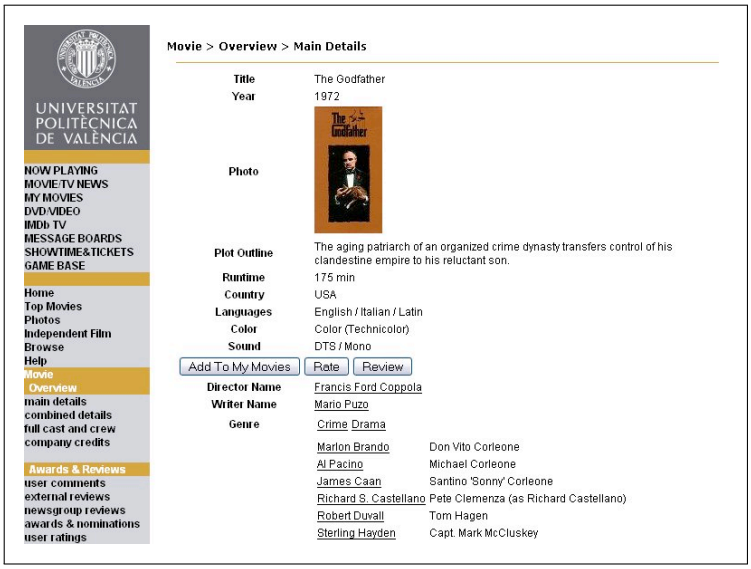


Figura C.17 – *IMDb Lite*: Pàgina Web *Movie.Overview.MainDetails (estil UPV)*

amb l'estil visual de la UPV.

Apèndix D

Un Compilador de Models d'OOWS

Aquest annexe presenta una implementació de les regles de transformació definides pel mètode OOWS al Capítol 8, segons es detalla al Capítol 9, emprant el llenguatge oAW Xpand [127]. És a dir, realitzar les transformacions de Model a Codi, sent l'entrada un model instància del metamodel d'OOWS en Ecore, i sent l'eixida el codi en un Framework d'Interfícies Web desenvolupat en PHP (vist a l'Annexe A).

Seguint l'estructura d'organització que proposa Xpand, cada regla de transformació es descriu mitjançant una sentència DEFINE. Però també permet organitzar aquestes en diferents fitxers o *paquets* com un mecanisme per simplificar i estructurar la definició d'aquestes regles.

D.1 Configuració Inicial Aplicació. Plantilla *Root*

La plantilla *Root* és la principal a l'hora d'aplicar les transformacions. És l'encarregada de crear la configuració inicial de l'aplicació (declaració

de pàgines i usuaris, estil de visualització, pàgina per defecte, etc.) i processar els usuaris (*UserRol*) existents, invocant la plantilla per generar els nodes navegacionals, obtenint així la implementació del mapa navegacional per a cadascun d'aquestos usuaris.

```
«IMPORT oows»
«EXTENSION template::helpers »

«DEFINE Root FOR OOWSModel-»
        «FILE id +".php"-»
<?php
include_once "../../OOWSFramework/ApplicationBegin.php";

        $Application = new Application("«id»","ONME","«id»");
        $Application->DefaultStyle("«id»");

        «FOREACH NavigationalModel.UserRol AS Rol -»
                «IF Rol.id == "Anonymous" -»
        $Application->AllowAnonymous();
                «ELSE -»
        $Application->Rol("«Rol.id»","«Rol.Class.id»","«Rol.Class.
            id»_MVAgentValidation","«Rol.Parent.id»");
                «ENDIF -»
                «EXPAND NavigationalNodes_rule FOREACH Rol.
                    NavigationalNode-»
        «ENDFOREACH -»
        $Application->HomePage("«NavigationalModel.Root.id»");

        «PROTECT CSTART "/*" CEND "*/" ID id»
        //Put your custom code here
        «ENDPROTECT»

include_once "../../OOWSFramework/ApplicationEnd.php";
?>
        «ENDFILE»
«ENDDEFINE»

«DEFINE NavigationalNodes_rule FOR NavigationalNode-»
«ENDDEFINE»


«DEFINE NavigationalNodes_rule FOR NavigationalContext-»
        $Application->Page("«id»","«alias»","«UserRol.id»","
            «reachability.Access()»");
        «EXPAND ContextTemplate::Context_Rules FOR this-»
«ENDDEFINE»


«DEFINE NavigationalNodes_rule FOR NavigationalSubsystem-»
        $Application->PageGroup("«id»","«UserRol.id»","Root");
        «EXPAND NavSubsystem_rule FOR this-»
        «EXPAND ContextTemplate::SubsystemPage_rule FOR this-»
```

```
«ENDDEFINE»


«DEFINE NavSubsystem_rule FOR NavigationalSubsystem-»
        «FOREACH NavigationalNode.typeSelect(NavigationalContext)
            AS navContext -»
        $Application->AddPageToGroup("«navContext.id»","
            «navContext.alias»","«id»","«navContext.reachability.
            Access()»");
        «EXPAND ContextTemplate::Context_Rules FOR navContext-»
        «ENDFOREACH -»

        «FOREACH NavigationalNode.typeSelect(NavigationalSubsystem
            ) AS navSubsystem -»
        $Application->PageGroup("«navSubsystem.id»","«UserRol.id»
            ","«id»");
        «EXPAND NavSubsystem_rule FOR navSubsystem-»
        «ENDFOREACH -»
«ENDDEFINE»
```

Codi Font D.1 – Plantilla de Transformació *Root* OOWSGenerator.xpt

Per cada node navegacional, s'aplica la seua plantilla en funció de si es tracta d'un Contexte Navegacional o d'un Subsistema.

La plantilla per als Nodes navegacionals de tipus Contexte, defineixen una pàgina Web (*Page*) i llancen l'execució de la plantilla associada a un contexte (veure Secció D.2).

La plantilla per als Nodes Navegacionals de tipus Subsistema, defineixen un grup de pàgines Web (*PageGroup*) i llancen l'execució de la plantilla associada a un subsistema (veure Secció D.2).

D.2 Transformació de Contextes i Subsistemes

Aquesta plantilla és possiblement la més important dins la transformació dels models ja que s'encarrega de realitzar la definició detallada del contingut de les pàgines.

Si es tracta d'un **contexte** navegacional, la seua plantilla n'és responsable de crear la pàgina Web i determinar les zones de contingut (*Zone*) que ha de tenir. Per a cada UAI definida al contexte, la plantilla

AIU_Rules agafa la informació de l'estructura de classes navegacionals per definir el contingut en base als atributs i relacions navegacionals. Per a cada relació navegacional, es llença l'execució de la plantilla associada (veure Secció D.3).

```
«IMPORT oows»
«DEFINE Context_Rules FOR NavigationalContext»
        «FILE "/Pages/" + id + ".php"-»
<?php
include_once "../../../OOWSFramework/PageBegin.php";

        $Page = new Page("«alias»");
        $Page->AddUserInfoZone("UserZone","Logged User");
        «EXPAND AIU_Rules FOREACH AIU-»

        «PROTECT CSTART "/*" CEND "*/" ID id »
        //Put your custom code here
        «ENDPROTECT»

include_once "../../../OOWSFramework/PageEnd.php";
?>
        «ENDFILE-»
«ENDDEFINE»

//Crea un pagina auxiliar con los enlaces a las paginas del
    subsistema
«DEFINE SubsystemPage_rule FOR NavigationalSubsystem»
        «FILE "/Pages/" + id + ".php"-»
<?php
include_once "../../../OOWSFramework/PageBegin.php";

        $Page = new Page("«alias»");
        $Page->AddUserInfoZone("UserZone","Logged User");

        $Zone= $Page->AddNavigationZone("«id»Menu","«alias» Menu")
            ;
        «FOREACH NavigationalNode AS childNode -»
        $Zone->NavigationLink("«childNode.id»","«childNode.alias»
            ","Root");
        «ENDFOREACH -»

        «PROTECT CSTART "/*" CEND "*/" ID id »
        //Put your custom code here
        «ENDPROTECT»

include_once "../../../OOWSFramework/PageEnd.php";
?>
        «ENDFILE-»
«ENDDEFINE»

«DEFINE AIU_Rules FOR AIU»
        //Main Information Zone
```

```
        $Zone = $Page->AddInformationZone("«id»","«id»","
            «ManagerClass.Class.id»");
        «FOREACH ManagerClass.NavigationalAttribute AS
            NavAttribute-»
        $Zone->Field("«NavAttribute.Attribute.ref»","«NavAttribute
            .alias»");
        «ENDFOREACH-»
        «EXPAND NavigationalRelationship::NavRelationship_rule(
            ManagerClass) FOREACH NavigationalRelationship -»

        //Services
        «EXPAND Service::Service_rule FOR this-»
        «REM» Si el AIU tiene una vista de busqueda asociada no
            recuperamos todas sus instancias de golpe «ENDREM»
        «IF SearchView.exists(e|e != null)-»
        $Zone->NotRetrieveAlways();
        «ENDIF-»
        //Mechanism access
        «IF ManagerClass.Pagination != null-»
                «EXPAND AccessMechanisms::Pagination_rule FOR
                    ManagerClass.Pagination-»
        «ENDIF»
        «EXPAND AccessMechanisms::OrderBy_rule FOREACH
            ManagerClass.OrderBy-»

        «EXPAND AccessMechanisms::Filter_rule FOREACH Filter-»

        «EXPAND AccessMechanisms::Index_rule FOREACH Index-»
«ENDDEFINE»
```

Codi Font D.2 – Plantilla de Transformació ContextTemplate.xpt

Si es tracta d'un **subsistema** navegacional es tracta aquest conjunt de pàgines com si d'un mapa navegacional: es defineix l'enllaç navegacional dins el node que representa el subsistema (*NavigationalLink*) i recursivament es crida a la plantilla per processar cada node, com s'ha vist a la secció anterior.

D.3 Transformació de les Relacions Navegacionals

La plantilla per transformar les relacions navegacional comença detectant el tipus de relació (reflexiva o relació navegacional) i llança l'execució de

les regles adequades.

Les regles per transformar aquestes **relacions navegacionals** (*RelationshipEnds_rule*) comproven les classes navegacionals origen i destí de la relació, així com la cardinalitat de la relació per incloure la informació relacionada correctament. En cas de tractar-se de **relacions navegacionals de tipus contexte**, es crea també un enllaç mitjançant l'operació del framework*InternalLinkTo*.

```
«IMPORT oows»
«DEFINE NavRelationship_rule (ManagerClass mc) FOR
    NavigationalRelationship-»
        «REM»Detectamos las relaciones reflexivas mediante
            this«ENDREM»
        «IF id == "this" -»
                «EXPAND NavigationLink_rule ("$Zone->") FOR this-»
        «REM»Determinamos si el detalle se creara a partir de la
            clase directora u otro detalle«ENDREM»
        «ELSEIF Source.NavigationalClass.Class.id == mc.Class.id -
            »
                «EXPAND RelationshipEnds_rule ("$Zone->") FOR this
                    -»
        «ELSE-»
                «EXPAND RelationshipEnds_rule ("$Detail"+ Source.
                    NavigationalClass.Class.id+"->") FOR this-»
        «ENDIF-»
«ENDDEFINE»


«DEFINE RelationshipEnds_rule (String pZoneId) FOR
    NavigationalRelationship-»
        «REM»Con cardinalidad 1 añadimos los atributos de las
            clases complementarias como relacionados«ENDREM»
        «IF Target.AssociationEnd.maxCardinality == "1" -»
                «EXPAND RelatedFields_rule (pZoneId) FOREACH
                    Target.NavigationalClass.NavigationalAttribute-
                    »
                «EXPAND NavigationLink_rule (pZoneId,Target.
                    NavigationalClass) FOR this -»
        «REM»Si la cardinalidad es N, es neceario añadir un
            detalle«ENDREM»
        «ELSE -»
                «LET Target.NavigationalClass.Class.id AS vDetail
                    -»
        $Detail«vDetail» = «pZoneId»DetailRelationship("«vDetail»
            ","«Source.AssociationEnd.role+"_"+Source.
            NavigationalClass.Class.id»","«alias»");
                «EXPAND Fields_rule ("$Detail" + vDetail + "->")
                    FOREACH Target.NavigationalClass.
                    NavigationalAttribute-»
```

```
                «EXPAND NavigationLink_rule ("$Detail" + vDetail +
                    "->") FOR this -»
                «ENDLET-»
«ENDIF-»
«ENDDEFINE»

//Reglas dummy para evitar problemas con la herencia
«DEFINE NavigationLink_rule (String pHeader) FOR
    NavigationalRelationship-»
«ENDDEFINE»

«DEFINE NavigationLink_rule (String pHeader,NavigationalClass
    pRelatedClass) FOR NavigationalRelationship-»
«ENDDEFINE»

//Añade el link de navegacion de las relaciones de contexto
«DEFINE NavigationLink_rule (String pHeader) FOR
    ContextRelationship-»
        «pHeader»«LinkAttribute.Attribute.ref»->InternLinkTo("
            «TargetContext.id»");
«ENDDEFINE»

«DEFINE NavigationLink_rule (String pHeader, NavigationalClass
    pRelatedClass) FOR ContextRelationship-»
        «pHeader»«pRelatedClass.Class.id»_«LinkAttribute.Attribute
            .ref»->InternLinkTo("«TargetContext.id»");
«ENDDEFINE»


«DEFINE Fields_rule (String pHeader) FOR NavigationalAttribute-»
        «pHeader»Field("«Attribute.ref»","«alias»");
«ENDDEFINE»


«DEFINE RelatedFields_rule (String pHeader) FOR
    NavigationalAttribute-»
        «pHeader»RelatedField("«Attribute.ref»","«alias»","
            «Attribute.Class.id»");
«ENDDEFINE»
```

Codi Font D.3 – Plantilla de Transformació NavigationalRelationship.xpt

D.4 Transformació dels Mecanismes d'Accés

Aquesta plantilla de regles de transformació s'executa a partir de les regles de transformació dels contextes navegacionals. Per a cada **Filtre** o **Índex** definit en un contexte, una regla concreta defineix el filtre o índex,

respectivament i construeix la vista de resultats (llençant l'execució de la regla *SearchView_ rule*).

Aquesta regla *SearchView_ rule* defineix les propietats de la vista de resultats (atributs, relacions, "atribut de selecció" i àlies) que s'emprarà en un índex o filtre.

```
«IMPORT oows»
«EXTENSION template::helpers »
«DEFINE Filter_rule FOR Filter ->»
«ENDDEFINE»


«DEFINE Filter_rule FOR StaticFilter->»
        «LET "f_"+id AS vFilterId ->»
        $«vFilterId» = $Zone->DefineStaticFilter("«vFilterId»","
            «description»","«Argument.Property()»","«operator»","
            «Value»");
        «IF SearchView != null->»
                «EXPAND SearchView_rule (vFilterId) FOR SearchView
                    ->»
        «ENDIF->»
        «ENDLET->»
«ENDDEFINE»


«DEFINE Filter_rule FOR DynamicFilter->»
        «LET "f_"+id AS vFilterId ->»
        $«vFilterId» = $Zone->DefineDynamicFilter("«vFilterId»","
            «description»","«Argument.Property()»","«operator»");
        «IF SearchView != null->»
                «EXPAND SearchView_rule (vFilterId) FOR SearchView
                    ->»
        «ENDIF->»
        «ENDLET->»
«ENDDEFINE»


«DEFINE Index_rule FOR Index ->»
        «LET "i_"+id AS vIndexId ->»
        $«vIndexId» = $Zone->DefineIndex("«vIndexId»","
            «description»","«Attribute.Property()»");
        «IF SearchView != null->»
                «EXPAND SearchView_rule (vIndexId) FOR SearchView-
                    »
        «ENDIF->»
        «ENDLET->»
«ENDDEFINE»


«DEFINE SearchView_rule (String pIdAccess) FOR SearchView ->»
        $«pIdAccess»->SearchView("«AIU.ManagerClass.Class.id»","
```

```
            «id»");
        $«pIdAccess»->SelectionAttribute("«SelectionAttribute.
            Attribute.ref»","«SelectionAttribute.alias»");
        «FOREACH SearchViewAttribute AS navAttribute->»
        $«pIdAccess»->Attribute("«navAttribute.Property()»","
            «navAttribute.alias»");
        «ENDFOREACH ->»
«ENDDEFINE»

«DEFINE Pagination_rule FOR Pagination ->»
        $Zone->SetPagination(«cardinality»);
«ENDDEFINE»

«DEFINE OrderBy_rule FOR OrderBy ->»
        $Zone->AddSortCondition("«NavigationalAttribute.Attribute.
            ref»","«direction.Order()»");
«ENDDEFINE»
```

Codi Font D.4 – Plantilla de Transformació AccessMechanisms.xpt

D.5 Transformació de l'accés a la Funcionalitat

Per cada operació que es detecte en un contexte navegacional dins una classe navegacional, es llança la regla *Service_rule*. Aquesta regla s'encarrega de crear una zona (*AddServicesZone*) que contindrà tota la informació per poder llançar l'execució de l'operació que existirà a la lògica de negoci de l'aplicació.

La regla *ServiceReference_rule* defineix la referència a l'operació del diagrama de classes[1], i construeix el formulari associat a l'operació (*ServiceForm_rule*).

```
«IMPORT oows»
«DEFINE Service_rule FOR AIU->»
        «IF ManagerClass.NavigationalOperation.exists(op|op!=null)
            ->»
                «LET ManagerClass.Class.id + "ServicesZone" AS
                    vZoneId ->»
        $«vZoneId» = $Page->AddServicesZone("«vZoneId»","
            «ManagerClass.Class.id +" Services"»");
```

[1]Aquesta operació serà llançada pel framework com es descriu a Annexe A.

```
                «EXPAND ServiceReference_rule (vZoneId) FOREACH
                    ManagerClass.NavigationalOperation -»
                «ENDLET-»
        «ENDIF-»

        «FOREACH ComplementaryClass AS compClass -»
        «IF compClass.NavigationalOperation.exists(op|op!=null)-»
                «LET compClass.Class.id + "ServicesZone" AS
                    vZoneId -»
        $«vZoneId» = $Page->AddServicesZone("«vZoneId»","
            «compClass.Class.id +" Services"»");
                «EXPAND ServiceReference_rule (vZoneId) FOREACH
                    compClass.NavigationalOperation -»
                «ENDLET-»
        «ENDIF-»
        «ENDFOREACH -»
«ENDDEFINE»


«DEFINE ServiceReference_rule (String pHeader) FOR
    NavigationalOperation -»
        $«pHeader»->ServiceReference("«Operation.id»","«alias»");
        «EXPAND  ServiceForm_rule (alias) FOR Operation-»
«ENDDEFINE»


«DEFINE ServiceForm_rule (String pAlias) FOR Operation-»
        «FILE "/Services/" + id + ".php" -»
<?php
include_once "../../../OOWSFramework/ServiceBegin.php";

        $Service = new Service("«pAlias»");

        «EXPAND Parameter_rule FOREACH Argument -»

        «PROTECT CSTART "/*" CEND "*/" ID id»
        //Put your custom code here
        «ENDPROTECT»

include_once "../../../OOWSFramework/ServiceEnd.php";
?>
        «ENDFILE -»
«ENDDEFINE»


«DEFINE Parameter_rule FOR Argument-»
        «IF dataType == DataTypes::Auto-»
        $Service->ArgNumber("«ref»","«id»");
        $Service->«ref»->Value(-1);
        «ELSEIF dataType == DataTypes::Int -»
        $Service->ArgNumber("«ref»","«id»");
        «ELSEIF dataType == DataTypes::Natural -»
        $Service->ArgNumber("«ref»","«id»");
        $Service->«ref»->SetNatural();
```

```
        «ELSEIF dataType == DataTypes::Real-»
        $Service->ArgReal("«ref»","«id»");
        «ELSEIF dataType == DataTypes::String-»
        $Service->ArgText("«ref»","«id»",32);
        «ELSEIF dataType == DataTypes::Text-»
        $Service->ArgText("«ref»","«id»",32);
        $Service->«ref»->Lines(4);
        «ELSEIF dataType == DataTypes::Bool-»
        $Service->ArgBool("«ref»","«id»");
        «ELSEIF dataType == DataTypes::Date-»
        $Service->ArgDate("«ref»","«id»");
        «ELSEIF dataType == DataTypes::Time-»
        $Service->ArgTime("«ref»","«id»");
        «ELSEIF dataType == DataTypes::password-»
        $Service->ArgPassword("«ref»","«id»",16);
        «ELSEIF dataType == DataTypes::Object-»
        $Service->ArgReference("«ref»","«id»","«ObjectValuatedTo.
            id»");
        $Service->«ref»->SetDynamicValues("«ObjectValuatedTo.id»
            ","id_«ObjectValuatedTo.id»","id_«ObjectValuatedTo.id»
            ");
        «ENDIF-»
«ENDDEFINE»
```

Codi Font D.5 – Plantilla de Transformació Service.xpt

La regla per crear el formulari d'introducció de dades, *Service-Form_ rule*, crea una nova pàgina de tipus *Service* (que respon al tipus de pàgina d'introducció de dades vistes al Capítol 8) i per a cada argument de l'operació crea un paràmetre al formulari en funció del tipus de dada associat a l'argument.

D.6 Extensions Auxiliars amb Extend

Emprant el concepte *Extend* de Xpand, en aquesta plantilla hi ha funcionalitat que s'utilitza per processar les regles de transformació anteriors, actuant a mode de llibreria auxiliar de funcions.

```
import oows;

//Transforma el tipo de contexto (Secuencia/Exploracion) del tipo
    enumerado a los valores
//que soporta el framework
```

```
String Access(Reachabilities r):  r == Reachabilities::Exploration
      ? "Always" : "FromPage";

//Concatena la clase con el atributo
String Property(NavigationalAttribute this): Attribute.Class.id
    +"."+Attribute.ref;

//Transforma el tipo de ordenacion a valores del framework;
String Order(DirectionTypes d):  d == DirectionTypes::Ascendant  ?
     "ASC" : "DESC";
```

Codi Font D.6 – Extensions auxiliars amb Extend

Referències

[1] S. Abrahao. *On the Functional Size Measurement of Object-Oriented Conceptual Schemas: Design and Evaluation Issues*. PhD thesis, Departament de Sistemes Informàticos i Computació (Universitat Politècnica de València), Valencia, Spain, October 2004.

[2] ACM. Special Issue on Personalization. *Communications of the ACM*, 43(8), 2000.

[3] AGG: Attributed Graph Grammar System v6.0. http://tfs.cs.tu-berlin.de/agg/, (última visita 20-12-2007).

[4] *Adaptive Hypermedia and Adaptive Web-Based Systems, 5th International Conference, AH 2008, Hannover, Germany, July 28 - August 1, 2008*, 2008.

[5] AJAX: Asynchronous JavaScript And XML.

[6] M. Albert, J. Muñoz, V. Pelechano, and O. Pastor. Model to Text Transformation in Practice: Generating Code From Rich Associations Specifications. In *BP-UML 2006 - Best Practices in UML, 2nd. International Workshop, Proceedings*, Tucson, USA, 2006.

[7] AMACONT Project: system Architecture for Multimedia Adaptive WebCONTent. http://www-mmt.inf.tu-

dresden.de/Forschung/Projekte/AMACONT/index.xhtml, (última visita 20-12-2007).

[8] Amazon.com Virtual eCommerce Shop. http://www.amazon.com.

[9] AndroMDA. http://www.andromda.org, (última visita 20-12-2007).

[10] Apache Software Foundation. The Apache Velocity Project. http://velocity.apache.org/, (última visita 20-12-2007).

[11] C. Atkinson and T. Kühne. Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5):36–41, 2003.

[12] ATLAS Group. ATL: ATLAS Transformation Language. http://www.eclipse.org/m2m/atl/, (última visita 20-12-2007).

[13] L. Baresi, F. Garzotto, and P. Paolini. From Web Sites to Web Applications: New Issues for Conceptual Modeling. In *ER 2000 - Conference on Conceptual Modeling, 19th International Conference, Proceedings*, volume 1921 of *Lecture Notes in Computer Science*, Salt Lake City, USA, 2000. Springer.

[14] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[15] A. Bonifati, S. Ceri, and P. Fraternali. Building Multi-Device, Content-Centric Applications Using WebML and the W3I3 Tool Suite. In *ER 2000 - Conference on Conceptual Modeling, 19th International Conference, Proceedings*, Salt Lake City, USA, 2000. Springer.

[16] Borland. Borland Together: Visual Modeling for Software Architecture Design.

http://www.borland.com/us/products/together/index.html, (última visita 20-12-2007).

[17] M. Brambilla, S. Ceri, S. Comai, P. Fraternali, and I. Manolescu. Specification and design of workflow-driven hypertexts. *JWE - Journal of Web Engineering, Rinton Press*, 1(2):163–182, April 2003.

[18] M. Brambilla, S. Ceri, M. Passamani, and A. Riccio. Managing Asynchronous Web Services Interactions. In *ICWS 2004 - Conference on Web Services, International Conference, Proceedings*, pages 80–87, San Diego, USA, 2004.

[19] M. Brambilla and C. Tziviskou. Fundamentals of exception handling within workflow-based web applications. *JWE - Journal of Web Engineering, Rinton Press*, 4(1):38–56, March 2005.

[20] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0, RDF Schema (W3C Recommendation). http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, February 2004.

[21] P. Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.

[22] CakePHP: Rapid Development Framework for PHP. http://www.cakephp.org, (última visita 20-12-2007).

[23] CARE Technologies S.A. OlivaNOVA. http://www.care-t.com/products/index.asp, (última visita 20-12-2007).

[24] CARE Technologies S.A. OlivaNOVA Modeller. http://www.care-t.com/products/modeler.asp, (última visita 20-12-2007).

[25] CARE Technologies S.A. OlivaNOVA Transformation Engines. http://www.care-t.com/products/transengine.asp, (última visita 20-12-2007).

[26] CARE Technologies S.A. http://www.care-t.com, (última visita 20-12-2007).

[27] S. Ceri, F. Daniel, M. Matera, and F. Facca. Model-driven Development of Context-Aware Web Applications. *ACM TOIT - ACM Transactions on Internet Technology*, (to appear), 2007.

[28] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. *Computer Networks*, 33:137–157, 2000.

[29] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. In *WWW 2000 - World Wide Web Conference, 9th International Conference, Proceedings*, pages 137–157. Elsevier, 2000.

[30] S. Ceri, P. Fraternali, and M. Matera. Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing*, 6(4):20–30, 2002.

[31] S. Ceri, M. Matera, F. Rizzo, and V. Demaldè. Designing data - intensive Web Applications for content accessibility using Web Marts. *Communications of the ACM*, 50(4):55–61, 2007.

[32] Code Generation Network. Code Generation Information for the Pragmatic Engineer: Quality, Consistency, Productivity and Abstraction. http://www.codegeneration.net/generators-by-standard.php?standard=1, (última visita 20-12-2007).

[33] Compuware. OptimalJ: Model-driven development for Java. http://www.compuware.com/products/optimalj, (última visita 20-12-2007).

[34] J. Conallen. *Building Web Applications with UML 2.* Addison-Wesley, 2003.

[35] O. De Troyer. Audience-driven Web Design. In *Information Modeling in the New Millennium.* Idea Group, 2001.

[36] O. De Troyer and S. Casteleyn. Modelling Complex Processes from web applications using WSDM. In *IWWOST 2003 - Web Oriented Software Technologies, Third International Workshop, Proceedings*, pages 1–12, Oviedo, Spain, 2003.

[37] O. De Troyer and C. Leune. WSDM: A User-centered Design Method for Web sites. In *WWW 1997 - World Wide Web Conference, 7th International Conference, Proceedings*, pages 85–94, 1997.

[38] Y. Desphande, S. Murugesan, and S. Hansen. Web Engineering: Beyond CS, IS and SE Evolutionary and Non-engineering Perspectives. In *Web Engineering: Managing Diversity and Complexity of Web Application Development, Proceedings*, volume 2016 of *Lecture Notes in Computer Science*, pages 14–23, 2001.

[39] Department of Information Systems and Computation, Web Site.

[40] Eclipse Foundation. Eclipse Modelling Framework. http://www.eclipse.org/modeling/emf/, (última visita 20-12-2007).

[41] Eclipse Foundation. Eclipse Modelling Project. http://www.eclipse.org/modeling/, (última visita 20-12-2007).

[42] Eclipse Foundation. GMF: Graphical Modeling Framework. http://www.eclipse.org/gmf/, (última visita 20-12-2007).

[43] Eclipse Foundation. MOFScript: Developing Tools and Frameworks for supporting Model to Text Transformations. http://www.eclipse.org/gmt/mofscript/, (última visita 20-12-2007).

[44] J. Fons, V. Pelechano, M. Albert, and O. Pastor. Development of Web Applications from Web Enhanced Conceptual Schemas. In *ER 2003 - Workshop on Conceptual Modeling and the Web, 22nd International Conference, Proceedings*, volume 2813 of *Lecture Notes in Computer Science*, Chicago, USA, 2003. Springer.

[45] F. Frasincar, G. Houben, and P. Barna. HPG: the Hera Presentation Generator. *JWE - Journal of Web Engineering, Rinton Press*, 5(2):175–200, 2006.

[46] P. Fraternali. Tools and approaches for developing data-intensive Web applications: a survey. *ACM Computing Surveys, ACM Press*, 31(3):227–263, September 1999.

[47] FreeMarker. http://freemarker.org/, (última visita 20-12-2007).

[48] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley, 1995.

[49] F. Garzotto, L. Mainetti, and P. Paolini. Hypermedia Design, Analysis and Evaluation Issues. *Communications of the ACM*, 38(8):74–86, 1995.

[50] F. Garzotto, P. Paolini, and D. Schwabe. HDM - A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, 11(1):1–26, 1993.

[51] GentleWare. Poseidon for UML. http://www.gentleware.com/, (última visita 20-12-2007).

[52] J. Gómez, C. Cachero, and O. Pastor. Extending a Conceptual Modeling Approach to Web Application Design. In *CAiSE 2000 - Conference on Advanced Information Systems Engineering, International Conference, Proceedings*, volume 1789 of *Lecture Notes in Computer Science*, pages 79–93, Interlaken, Suiza, 2000. Springer.

[53] D. Harel. *Handbook of Philosophical Logic II*, chapter 10 Dynamic Logic, pages 497–694. D.M. Gabbay and F. Guenthner, Reidel, 1984.

[54] R. Hennicker and N. Koch. A UML-based Methodology for Hypermedia Design. In A. Evans, S. Kent, and B. Selic, editors, *UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, Proceedings*, pages 410–424, York, United Kingdom, 2000. Springer.

[55] G. Houben, P. Barna, F. Frasincar, and R. Vdovjak. Hera: Development of Semantic Web Information Systems. In *ICWE 2003 - Web Engineering, Third International Conference, Proceedings*, volume 2722 of *Lecture Notes in Computer Science*, pages 529–538, Oviedo, Spain, July 2003. Springer.

[56] IBM. Rational Rose. http://www.ibm.com/software/awdtools/developer/rose/index.html, (última visita 20-12-2007).

[57] IBM High-Volume Web site team. Web Site Personalization. Technical report, IBM, January 2000.

[58] IMDb: The Internet Movie Database. http://www.imdb.org/, (última visita 20-12-2007).

[59] E. Insfrán, O. Pastor, and R. Wieringa. Requirements Engineering-Based Conceptual Modelling. *Requirements Engineering*, 7(2):61–72, 2002.

[60] Interactive Objects Software GmbH. ArcStyler: The Dynamic Link Between Business and Technology. http://www.interactive-objects.com/products/arcstyler, (última visita 20-12-2007).

[61] Jack Greenfield and Keith Short and Steve Cook and Stuart Kent. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley, September 2004.

[62] G. Kappel, W. Retschitzegger, and W. Schwinger. Modeling Customizable Web Applications - A Requirement's Perspective. In *Digital Libraries: Research and Practice, International Conference, Proceedings*, Kyoto, Japan, November 2000.

[63] A. Knapp, N. Koch, F. Moser, and G. Zhang. ArgoUWE: A CASE Tool for Web Applications. In J. Ralyte and C. Roland, editors, *EMSISE 2003 - Engineering Methods to Support Information Systems Evolution, 1st International Workshop, Proceedings*, pages 37–50, Geneve, France, 2003.

[64] A. Knapp, G. Zhang, and H. Hassler. Modeling Business Processes in Web Applications with ArgoUWE. In T. Baar, A. Strohmeier, A. Moreira, and S. Mellor, editors, *UML 2004 - The Unified Modeling Language.Modelling Languages and Applications, 7th International Conference, Proceedings*, volume 3273 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.

[65] N. Koch. *Software Engineering for Adaptive Hypermedia Applications*. PhD thesis, Lüdwig-Maximilians-University, Munich, Germany, 2000.

[66] N. Koch, S. Meliá, N. Moreno, V. Pelechano, F. Sánchez-Figueroa, and J. Vara. Model-Driven Web Engineering. *Novática-Upgrade. Special Issue on MDA at the Age of Seven*, (en espera de publicació), March/April 2008.

[67] N. Koch and M. Wirsing. Software Engineering for Adaptive Hypermedia Applications. In *Workshop on Adaptive Hypertext and Hypermedia, 3rd International Edition*, Málaga, Spain, 2001.

[68] N. Koch and M. Wirsing. The Munich Reference Model forAdaptive Hypermedia Applications. In P. De Bra, P. Brusilovsky, and R. Conejo, editors, *AH 2002 - Adaptive Hypermedia and Adaptive Web Based Systems, Second International Conference, Proceedings*, volume 2347 of *Lecture Notes in Computer Science*, pages 213–222, Malaga, Spain, 2002. Springer.

[69] J. Kovse and T. Härder. Generic XMI-Based UML Model Transformations. In *Object-Oriented Information Systems, 8th International Conference, Proceedings*, Montpellier, France, 2002.

[70] H. Lieberman, F. Paternò, and V. Wulf, editors. *End User Development*, volume 9. Springer, 2006.

[71] F. Lima and D. Schwabe. Modeling Applications for the Semantic Web. In *ICWE 2003 - Web Engineering, Third International Conference, Proceedings*, volume 2722 of *Lecture Notes in Computer Science*, pages 417–426, Oviedo, Spain, July 2003. Springer.

[72] I. Manolescu, M. Brambilla, S. Ceri, S. Comai, and P. Fraternali. Model-Driven Design and Deployment of Service-Enabled Web Applications. *ACM TOIT - ACM Transactions on Internet Technology*, 5(3):439–479, August 2005.

[73] B. Marín, G. Giachetti, and O. Pastor. ntercambio de Modelos UML y OO-Method. In *IDEAS 2007 - Ingeniería de Requisitos y Ambientes Software, 10th Iberoamerican Workshop, Proceedings*, pages 283–296, Isla Margarita, Venezuela, 2007.

[74] S. Meliá and C. Cachero. An MDA Approach for the Development of Web Applications. In N. Koch, P. Fraternali, and M. Wirsing, editors, *ICWE 2004 - Web Engineering, 4th International Conference, Proceedings*, volume 3140 of *Lecture Notes in Computer Science*, pages 300–305, Munich, Germany, July 2004. Springer.

[75] S. Meliá, A. Kraus, and N. Koch. MDA Transformations Applied to Web Application Development. In D. Lowe and M. Gaedke, editors, *ICWE 2005 - Web Engineering, 5th International Conference, Proceedings*, volume 3579 of *Lecture Notes in Computer Science*, pages 465–471, Sydney, Australia, 2005. Springer.

[76] S. Mellor, A. Clark, and T. Futagami. Guest Editors' Introduction: Model-Driven development. *IEEE Software*, 20(5):14–18, 2003.

[77] Microsoft. ASP .NET: Active Server Pages .NET. http://www.asp.net/, (última visita 20-12-2007).

[78] Microsoft. DSL Tools. http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx, (última visita 20-12-2007).

[79] J. Muñoz and V. Pelechano. Building a Software Factory for Pervasive Systems Development. In O. Pastor and J. Falcão e Cunha, editors, *CAiSE 2005 - Conference on Advanced Information Systems Engineering, 17th International Conference, Proceedings*, volume 3520 of *Lecture Notes in Computer Science*, pages 342–356, Porto, Portugal, 3-540-26095-1 2005. Springer.

[80] S. Murguesan, Y. Desphande, and A. Ginigie. Web Engineering: A New Discipline for Development of Web-based Systems. In *Web Engineering (ICSE) 1999, International Workshop, Proceedings (within the International Conference on Software Engineering)*, 1999.

[81] S. Murugesan and Y. Desphande. Web Engineering. *Software Engineering and Web Application Development. Springer LNCS - HotTopics*, 2001.

[82] No Magic, Inc. MagicDraw: Architecture Made Simple. http://www.magicdraw.com/, (última visita 20-12-2007).

[83] D. Nunes and D. Schwabe. Rapid Prototyping of Web Applications combining Domain Specific Languages and Model Driven Design. In D. Wolber, N. Calder, C. Brooks, and A. Ginige, editors, *ICWE 2006 - Web Engineering, 6th International Conference, Proceedings*, pages 153–160, Palo Alto, USA, 2006. ACM Press.

[84] Object Management Group, OMG. Unified Modeling Language (UML) Specification Version 1.4 draft. Technical report, www.omg.org, February 2001.

[85] Object Management Group, OMG. MOF2Text: Model to Text Transformation Language, Request for Proposals. http://www.omg.org/cgi-bin/doc?ad/2004-4-7, May 2004 (última visita 20-12-2007).

[86] Object Management Group, OMG. Model Driven Architecture (MDA). http://www.omg.org/mda, 2006 (última visita 20-12-2007).

[87] Object Management Group, OMG. MetaObject Facility. http://www.omg.org/mof/, (última visita 20-12-2007).

[88] L. Olsina. *Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web.* PhD thesis, Facultad de Ciencias Excatas de la Universidad Nacional de La Plata, 1999. In spanish.

[89] O. Pastor, J. Fons, S. Abrahão, and S. Ramón. Object-Oriented Conceptual Models for Web Applications. In *IDEAS 2001 - Ingeniería de Requisitos y Ambientes Software, 4th Iberoamerican Workshop, Proceedings*, San José, Costa Rica, 2001.

[90] O. Pastor, J. Fons, and V. Pelechano. A Method to Develop Web Applications from Web-Oriented Conceptual Models. In *IWWOST 2003 - Web Oriented Software Technologies, Third International Workshop, Proceedings*, pages 65–70, Oviedo, Spain, July 2003.

[91] O. Pastor, J. Fons, V. Torres, and V. Pelechano. Conceptual Modelling versus Semantic Web: the two sides of the same coin? In *Worshop on Application Design 2004 (WWW), International Workshop, Proceedings (withing the World Wide Web Conference)*, New York, USA, 2004.

[92] O. Pastor, J. Gómez, and C. Cachero. Conceptual Modeling of Device-Independent Web Applications. *IEEE Multimedia*, April-June 2001.

[93] O. Pastor, J. Gómez, E. Insfrán, and V. Pelechano. The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. *Information Systems*, 26(7):507–534, 2001.

[94] O. Pastor, E. Insfrán, V. Pelechano, J. Romero, and J. Merseguer. OO-Method: An OO Software Production Environment Combining Conventional and Formal Methods. In *CAiSE 1997*

- *Conference on Advanced Information Systems Engineering, 9th Edition, Proceedings*, volume 1250 of *Lecture Notes in Computer Science*, pages 145–159, Barcelona, Spain, June 1997. Springer.

[95] O. Pastor and J. Molina. *Model-Driven Architecture in Practice. A Software Production Environment Based on Conceptual Modeling.* Springer, 2007.

[96] F. Paternò. *ConcurTaskTrees: An Engineered Notation for Task Models*, chapter 24, pages 483–501. The Handbook of Task Analysis for Human-Computer Interaction. Lawrece Erlbaum Associates, London, United Kingdom, 2004.

[97] V. Pelechano. *Tratamiento de Relaciones Taxonómicas en Entornos de Producciõn Automática de Software. Una Aproximación Basada en Patrones.* PhD thesis, Departament de Sistemes Informàticos i Computació, Universitat Politècnica de València, València, Spain, 2001.

[98] T. Powell. *Web Site Engineering: Beyond Web Page Design.* Prentice Hall, 1998.

[99] R. Pressman. Can internet-based applications be engineered? *IEEE Software*, September/October, 1998.

[100] T. Reenskaug. The Model-View-Controller (MVC): Its Past and Present. Technical report, University of Oslo, 2003.

[101] G. Rojas. *Modelling Adaptive Web Applications in OOWS.* PhD thesis, Departament de Sistemes Informáticos i Computació, Valencia, Spain, 2008.

[102] G. Rojas and V. Pelechano. A methodological approach for incorporating adaptive navigation techniques into Web applications. In

WISE 2005 - Web Information Systems Engineering, Sixth International Conference, Proceedings, New York, USA, 2005.

[103] G. Rojas, V. Pelechano, and J. Fons. A model-driven approach to include adaptive navigational techniques in Web applications. In *IWWOST 2005 (CAiSE) - Web-Oriented Software Technologies, Fifth International Workshop, Proceedings (within the 17th Conference on Advanced Information Systems Engineering)*, Porto, Portugal, 2005.

[104] G. Rojas, P. Valderas, and V. Pelechano. Describing adaptive navigation requirements of Web applications. In *AH 2006 - Adaptive Hypermedia and Adaptive Web-Based Systems, Fourth International Conference, Proceedings*, Dublin, Ireland, 2006.

[105] G. Rossi. *Uma Metodologia Orientada a Objetos para o Projeto de Aplicativos Hipermidia*. PhD thesis, Departamento de Informática da PUC-Rio, Rio de Janeiro, Brasil, 1996.

[106] G. Rossi, F. Lyardet, and R. Guimarães. Designing Personalized Web Applications. In *WWW 2001 - World Wide Web Conference, 10th International Conference, Proceedings*, pages 275–284, Hong Kong, China, 2001. ACM Press.

[107] G. Rossi, F. Lyardet, and D. Schwabe. Patterns for E-Commerce Applications. In *EuroPloP 2000 - Pattern Languages of Programs, International Conference, Proceedings*, Kloster Irsee, Germany, 2000.

[108] G. Rossi and D. Schwabe. Object-Oriented Web Applications Modeling. In *Information Modeling in the New Millennium*, pages 463–484. Idea Group, 2001.

[109] G. Rossi and D. Schwabe. *Modelling and Implementing Web Applications with OOHDM*, chapter 6, pages 109–155. Web Engineering: Modelling and Implementing Web Applications. Human Computer Interaction Series. Gustavo Rossi and Oscar Pastor and Daniel Schwabe and Luis Olsina, eds. Springer, 2007.

[110] G. Rossi, D. Schwabe, and F. Lyardet. Web Application Models are More than Conceptual Models. In *ER 2000 - Conference on Conceptual Modeling, 19th International Conference, Proceedings*, Salt Lake City, USA, 2000. Springer-Verlag.

[111] G. Rossi, D. Schwabe, L. Olsina, and O. Pastor. *Overview of Design Issues for Web Applications Development*, chapter 4. Web Engineering: Modelling and Implementing Web Applications. Human Computer Interaction Series. Gustavo Rossi and Oscar Pastor and Daniel Schwabe and Luis Olsina, eds. Springer, 2007.

[112] G. Rozenberg. *Handbook of Graph Grammars and Computing by Graph Transformations*. World Scientific Publishing Company, 1997.

[113] Ruby on Rails: Web development that doesn't hurt. http://www.rubyonrails.org/, (última visita 20-12-2007).

[114] M. Ruiz, V. Pelechano, and O. Pastor. Designing Web Services for supporting user tasks: A model driven approach. In *CoSS 2006 - Conceptual Modeling of Service Oriented Software Systems, International Workshop, Proceedings*, 2006.

[115] M. Ruiz, P. Valderas, and V. Pelechano. Applying a Web Engineering method to design Web Services. In *ICSOC 2005 - Service-Oriented Computing, Third International Conference, Proceedings*, pages 576–581, 2005.

[116] M. Ruiz, F. Valverde, and V. Pelechano. Desarrollo de Servicios Web en un Mẽtodo de Generaciõn de Cõdigo Dirigido por Modelos. In *JSWEB 2006 - Jornadas Cientĩfico-Tẽcnicas en Servicios Web, II Edition, Proceedings*, 2006.

[117] D. Schwabe and G. Rossi. An object-oriented approach to Web-based application design. *Theory and Practice of Object Systems*, 4(4):207–225, October 1998.

[118] D. Schwabe and G. Rossi. An Object Oriented approach to Web-based applications design. *Theory and Practice of Object Systems*, 4(4):207–225, 1998.

[119] D. Schwabe and G. Rossi. Developing Hypermedia Applications Using OOHDM. In *Hypertext 1998 - Workshop on Hypermedia Development Processes, Methods and Models, Proceedings*, Pittsburgh, USA, 1998.

[120] D. Schwabe, G. Rossi, and S. Barbosa. Systematic Hypermedia Design with OOHDM. In *Hypertext 1996 - Workshop on Hypermedia Development Processes, Methods and Models, Proceedings*, Washington, USA, 1996.

[121] D. Schwabe, G. Szundy, S. S. de Moura, and F. Lima. Design and Implementation of Semantic Web Applications. In C. Bussler, S. Decker, D. Schwabe, and O. Pastor, editors, *Workshop on Application Design, Development and Implementations Issues in the Semantic Web 2004, Proceedings (within the World Wide Web Conference)*, volume 105, 2004.

[122] B. Selic. The Pragmatics of Model-Driven Development. *IEEE Software*, 20(5):19–25, 2003.

[123] S. Silva de Moura and D. Schwabe. Interface Development for Hypermedia Applications in the Semantic Web. In *WebMedia/LA-WEB 2004 - Latin American Web Congress, 2nd Edition, Proceedings (within the 10th Brazilian Symposium on Multimeda and the Web)*, pages 106–113, Ribeirao Preto-SP, Brazil, October 2004. IEEE Computer Society<.

[124] Sparx Systems. Enterprise Architect: The Advanced Software Modelling Tool for UML. http://www.sparxsystems.com.au/, (última visita 20-12-2007).

[125] Stanford Center for Biomedical Informatics Research. Protégé: a free, open source Ontology Editor and Knowledge base Framework. http://protege.stanford.edu/, (última visita 20-12-2007).

[126] The Apache Software Foundation. Apache Struts: Extensible framework for creating enterprise-ready Java web applications. http://struts.apache.org/, (última visita 20-12-2007).

[127] The oAW Team. oAW: The leading platform for professional model-driven software development. http://www.openarchitectureware.org/, (última visita 20-12-2007).

[128] The OO-Method Group. The OOWS Project. http://oomethod.dsic.upv.es/OOWS/.

[129] V. Torres and V. Pelechano. Building business process driven Web applications. In *BPM 2006 - Business Process Management, Fourth International Conference, Proceedings*, pages 322–337, Vienna, Austria, 2006.

[130] V. Torres, V. Pelechano, and O. Pastor. Building Semantic Web Services based on a model driven Web Engineering method. In

CoSS 2006 - Conceptual Modeling of Service Oriented Software Systems, International Workshop, Proceedings, Tucson, USA, 2006.

[131] P. Valderas. *A Requirements Engineering Approach for the Development of Web Applications.* PhD thesis, Departament de Sistemes Informàticos i Computació (Universitat Politècnica de València), 2008.

[132] P. Valderas, J. Fons, M. Albert, and V. Pelechano. Desarrollo de un portal Web para un Departamento Universitario desde Modelado Conceptual. In *IDEAS 2003 - Ingeniería de Requisitos y Ambientes Software, 6th Iberoamerican Workshop, Proceedings*, pages 15–26, Asunción, Paraguay, May 2003.

[133] P. Valderas, J. Fons, and V. Pelechano. Transforming Web Requirements into Navigational Models: An MDA Based Approach. In L. Delcambre, C. Kop, H. Mayr, J. Mylopoulos, and O. Pastor, editors, *ER 2005 - Workshop on Conceptual Modeling and the Web, 24th International Conference, Proceedings*, volume 3716 of *Lecture Notes in Computer Science*, pages 320–336, Klagenfurt/Velden, Austria, 2005. Springer.

[134] P. Valderas, V. Pelechano, and O. Pastor. A transformational approach to produce Web application prototypes from a Web requirements model. *International Journal on Web Engineering and Technology, IJWET*, 2006.

[135] P. Valderas, V. Pelechano, and O. Pastor. Towards and End-User development approach for Web Engineering Methods. In *CAiSE 2003 - Conference on Advanced Information Systems Engineering, 15th International Conference, Proceedings*, volume 4001 of *Lecture Notes in Computer Science*, pages 528–543, Luxembourg, June, 5-9 2006. Springer.

[136] A. Vallecillo, N. Koch, C. Cachero, S. Comai, P. Fraternali, I. Garrigós, J. Gómez, G. Kappel, A. Knapp, M. Matera, S. Meliá, N. Moreno, B. Pröll, T. Reiter, W. Retschitzegger, J. Rivera, A. Schauerhuber, W. Schwinger, M. Wimmer, and G. Zhang. MDWEnet: A Practical Approach to Achieving Interoperability of Model-Driven Web Engineering Methods (Position Paper). In *MDWEnet Worshop 2007 (within the ICWE 2007 - Web Engineering, 7th International Conference)*, pages 246–254, Como, Italy, July 2007.

[137] F. Valverde, P. Valderas, J. Fons, and V. Pelechano. A MDA-based Environment for Web Applications Development: From Conceptual Models to Code. In M. Brambilla and E. Mendes, editors, *IWWOST 2007 - Workshop on Web Oriented Software Technology, 7th International Edition, Proceedings (within the ICWE Conference)*, pages 164–178, 2007.

[138] K. van der Sluijs, G. Houben, J. Broekstra, and S. Casteleyn. Hera-S: Web Design using Sesame. In D. Wolber, N. Calder, C. Brooks, and A. Ginige, editors, *ICWE 2006 - Web Engineering, 6th International Conference, Proceedings*, pages 337–344, Palo Alto, USA, 2006. ACM Press.

[139] R. Vdovjak, F. Frasincar, G. Houben, and P. Barna. Engineering Semantic Web Information Systems in Hera. *JWE - Journal of Web Engineering, Rinton Press*, 1(2):3–26, 2003.

[140] P. Vilain, D. Schwabe, and C. Sieckenius de Souza. A Diagrammatic Tool for Representing User Interaction in UML. In A. Evans, S. Kent, and B. Selic, editors, *UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, Proceedings*, volume 1939 of *Lecture Notes in Computer Science*, pages 133–147, York, United Kingdom, October 2000. Springer.

[141] W3C Consortium. CSS: Cascade Style Sheets. http://www.w3.org/Style/CSS/, 1996.

[142] W3C Consortium. XHTML 1.0: eXtensible Hypertext Markup Language. A reformulation of HTML 4 in XML 1.0. W3C Recommendation. http://www.w3.org/MarkUp/, August 2002.

[143] W3C Consortium. RDF: Resource Description Framework. http://www.w3.org/RDF/, (última visita 20-12-2007).

[144] V. Wade, H. Ashman, and B. Smyth, editors. *Adaptive Hypermedia and Adaptive Web-Based Systems, 4th International Conference, AH 2006, Dublin, Ireland, June 21-23, 2006, Proceedings*, volume 4018 of *Lecture Notes in Computer Science*. Springer, 2006.

[145] WebML: Web Modelling Language. http://www.webml.org, (última visita 20-12-2007).

[146] WebRatio: You Think it, You get it. http://www.webratio.com, (última visita 20-12-2007).

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA