UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA
Facoltà di Ingegneria - Sede di Modena
Corso di Laurea in Ingegneria Informatica

Modellazione concettuale di interfacce web

Relatore
Chiar.mo Prof. Sonia Bergamaschi

Tesi di Laurea di
Silvia Balugani

Correlatore
Dott. Ing. Maurizio Vincini

Controrelatore
Chiar.mo Prof. Paolo Tiberio

Anno Accademico 2002 - 2003

Parole chiave:
Applicazioni Web
Modellazione concettuale
Front-end ipertestuale

RINGRAZIAMENTI

Desidero ringraziare la prof.ssa Bergamaschi e tutti coloro che hanno creduto in me e mi hanno aiutato in questo periodo così intenso

Contents

List of Figures

List of Tables

Introduzione

La crescita esponenziale e la capillare diffusione del web hanno introdotto una nuova generazione di applicazioni , caratterizzate da una relazione diretta tra il business ed il cliente.
Le *applicazioni web* sono divenute ormai un ingrediente fondamentale delle nostre attività lavorative,sociali e di relazione.
Tali applicazioni vengono oggi definite **Data-Intensive Web Applications**, infatti oggi il web è sempre più usato come piattaforma per sistemi informativi complessi,dove un'enorme quantità di dati soggetti a continui cambiamenti è gestita dai DBMS sottostanti. Le nuove aree di applicazione sono domini come il commercio elettronico, siti web di organizzazioni private e pubbliche, le librerie digitali ed i corsi a distanza. Le **interfacce web**(interfacce HTTP) infatti sono spesso considerate come lo strumento più potente per un' efficace presentazione e diffusione dell'informazione in quanto la combinazione di testi,suoni,video ed immagini permette diverse rappresentazioni dei concetti e l'uso di links rende naturale e flessibile l'accesso all'informazione.

Lo sviluppo delle *applicazioni web* è sempre più complicato, risulta dunque fondamentale una corretta **modellazione** nell'ambito del ciclo di vita di tali applicazioni per aumentarne la comprensione e facilitare la comunicazione all'interno del team di progetto.

Oggi i progettisti modellano le *applicazioni web* sulla base della loro conoscenza individuale,dunque applicando metodologie standard utilizzate nella modellazione delle applicazioni tradizionali, come le applicazioni object-oriented. Tali metodologie sono adatte per modellare la parte "convenzionale" della modellazione delle *web applications*,come il design delle strutture dati e della logica di business; ma risultano inadeguate per modellare ciò che caratterizza tali applicazioni: visualizzare contenuti e mettere a disposizione servizi utilizzando un **front-end ipertestuale**, cioè un **Web browser**. La modellazione di tali interfacce web segue infatti dei patterns e delle regole abbastanza consolidate nella pratica ma scarsamente dcumentate e supportate dagli strumenti di design tradizionali e dai linguaggi di modellazione standard.

Attualmente è forte l'esigenza di trovare una notazione chiara ed efficace per rappresentare le funzionalità dei componenti front-end dello strato web e la complessità della loro iterazione ed integrazione,allo scopo di generare diagrammi e documenti di specifica che permettano di ragionare ad un alto livello di astrazione senza vincolarsi a dettagli architetturali ed implementativi.

Lo scopo principale di questa tesi è: ricercare, analizzare e confrontare i linguaggi e formalismi attualmente proposti per la modellazione di interfacce web, per valutarne lo stato dell'arte ;dopo aver inizialmente individuato i componenti front-end(sia server-side che client-side) nell'ambito di una tipica architettura multilivello di un'applicazione distribuita, per comprendere effettivamente cosa si ha la necessità di modellare.
Obiettivo della tesi è studiare in modo approfondito un particolare linguaggio di modellazione web con applicazione ad un caso pratico , possibilmente utilizzando l'eventuale CASE tool

supportato da tale linguaggio ed arrivando a produrre la documentazione formale del design di una interfaccia.

La tesi è organizzata nel seguente modo:

Capitolo 1
In questo capitolo vengono descritti il ruolo della modellazione nella fase di progettazione e nell'utilizzo dei CASE tools per la produzione di un componente software e gli standard attualmente usati per la modellazione concettuale nei vari ambiti del software.Inoltre,viene descritta l'architettura multilivello di un tipico sistema informativo distribuito con particolare risalto all'individuazione dei componenti dello strato front-end.

Capitolo 2
In questo capitolo vengono individuati i requisiti fondamentali per un linguaggio di modellazione web,proponendo due punti di vista differenti per la suddivisione di tali requisiti in categorie.

Una prima categorizzazione prevede la suddivisione dei requisiti sulla base dei due aspetti dell'architettura tecnica delle applicazioni web : l'*architettura dell'informazione* e l'*architettura funzionale*.
Si possono individuare tre categorie:

- Requisiti legati all'informazione:come viene gestita e strutturata e l'accesso ad essa;
- Requisiti legati alle funzionalità del sistema;
- Requisiti di carattere generale .

La seconda categorizzazione prevede la suddivisione dei requisiti sulla base delle tre dimensioni ortogonali da considerare nella

modellazione delle applicazioni web:*livelli*,*aspetti*e *fasi*.

Capitolo 3
In questo capitolo viene studiato lo stato dell'arte dei linguaggi proposti per la modellazione di interfacce web,tali linguaggi vengono analizzati e confrontati tra loro sulla base dei requisiti indicati nel capitolo precedente.

Capitolo 4
In questo capitolo vengono descritti gli strumenti di sviluppo web *model-driven*;in particolare vengono descritti,applicati ad un caso pratico e confrontati due esempi significativi di CASE tools model-driven: *WebRatio Site Development Studio*,concepito al Politecnico di Milano e basato su un relativo linguaggio di modellazione(WebML),e *Rational Rose XDE* ,la versione del popolare CASE tool *Rational Rose* che fornisce alcuni profili UML per la modellazione Web adottando una particolare proposta di estensione di UML per il Web(UML esteso di Conallen).

Chapter 1

La modellazione concettuale dei componenti front-end

1.1 Il ruolo della modellazione nella fase di progettazione e nell'utilizzo dei CASE tool per la produzione di un componente software

Nella fase di progettazione nel ciclo di vita di un componente software è fondamentale modellare il contesto applicativo già delineato nelle specifiche dei requisiti.
La ***modellazione*** è una parte centrale delle attività che permettono lo sviluppo di un buon software (che soddisfa le necessità degli utenti) , in quanto aiuta a comprendere e gestire sistemi complessi.Nella fase di *modellazione* viene creata una documentazione formale della struttura e delle interrelazioni di un sistema facendo uso di diagrammi che esprimono modelli utilizzando notazioni grafiche.
L'uso di modelli rende espliciti i requisiti di progetto,realizza una migliore comunicazione tra i membri del team eliminando le ambiguità del linguaggio naturale, diminuisce i tempi di sviluppo ma sopratutto rende più chiari e mantenibili i documenti di design in quanto permette di ragionare ad un alto livello di as-

trazione senza essere vincolati a dettagli implementativi.I modelli aiutano a comprendere il sistema semplificando alcuni dettagli ; la scelta di cosa modellare è dunque fondamentale per l'analisi del problema e l'individuazione della soluzione.

Modellare il contesto applicativo di un prodotto software significa analizzare il sistema da tre punti di vista:

- Statico : si analizza e modella la realtà da inserire nel sistema ad un certo istante e si modella il database che conterrà i dati di interesse.

- Dinamico : si analizza e modella il comportamento dinamico del sistema, cioè l'evoluzione nel tempo delle entità che lo compongono e le relazioni tra esse.

- Funzionale : si analizza e modella il sistema dal punto di vista delle funzioni che deve mettere a disposizione,si tratta dell'aspetto più percettibile dall'utente finale.

Nell'ambito della modellazione si distingue tra *linguaggio di modellazione* e *metodologia*:

Un *linguaggio di modellazione* è la *notazione*(aspetto grafico dei modelli) usata nelle metodologie per esprimere le caratteristihe di un progetto.

Una *metodologia* è costituita da un linguaggio di modellazione e da *processo* che è una serie di consigli riguardanti le varie fasi della produzione del progetto. Naturalmente il linguaggio di modellazione è la parte più importante della metodologia, infatti modellando il sistema allo scopo di ottenere una comunicazione meno ambigua, è fondamentale la conoscenza del linguaggio comune utilizzato.

La creazione di modelli può attenersi più o meno strettamente alle notazioni del linguaggio di modellazione a seconda dello scopo per il quale ci si propone di utilizzare tali modelli.Se si ha a disposizione un **Computer-Aided Software Engineering(CASE)tool** per la generazione di codice,nella fase di modellazione bisogna attenersi strettamente all'interpretazione fornita dallo strumento per ottenere codice accettabile; se invece si utilizza la modellazione per la comunicazione lo sviluppo è più libero.
Naturalmente i benefici della modellazione vengono moltiplicati se sono disponibili tools adeguati che supportano i linguaggi di modellazione utilizzati.

1.2 La modellazione concettuale e le applicazioni web

Nell'ambito della modellazione vengono utilizzate notazioni con livelli di astrazione diversi:

- A *livello concettuale* viene effettata una rappresentazione astratta del sistema usando primitive di alto livello;
- A *livello logico* vengono utilizzate rappresentazioni di livello più basso vicine alle necessità dell'implementazione ,ma ancora indipendenti dalla tecnologia utilizzata;
- A *livello fisico* viene effettuato un design dipendente dalla tecnologia e strettamente legato a dettagli implementativi.

La ***modellazione concettuale*** viene utilizzata con successo in molti campi del software,in quanto permette di ragionare ad un alto livello di astrazione senza essere vincolati a dettagli architetturali o di implementazione.
Nella modellazione dei database ,ad esempio,il modello *Entity-Relationship*(*E/R*) [1] fornisce una notazione intuitiva e di alto

livello per la comunicazione dei requisiti legati ai dati tra i designers e persone non-tecniche ed è la base per creare schemi di database di alta qualità.
Più recentemente,con il crescere della complessità e delle dimensioni delle applicazioni software è nata la necessità di ottimizzare non tanto il prodotto,cioè la singola applicazione,quanto il processo di produzione,cioè il modo in cui un'applicazione viene concepita,progettata,realizzata e verificata. Nel settore dei sistemi informatici questo mutamento si è accellerato negli anni Novanta grazie al successo delle metodologie di sviluppo ad oggetti.
In particolare si è affermata l'idea che le fasi iniziali dello sviluppo di un'applicazione,l'analisi dei requsiti e la progettazione,fossero cruciali e dovessero essere sostenute da opportune notazioni formali e da adeguati strumenti software. La caratteristica comune delle diverse metodologie proposte è la ***modellazione concettuale*** basata su linguaggi visuali semplici ed intuitivi,ma al tempo stesso rigorosi e a volte formali.Grazie all'uso della modellazione concettuale è possibile generare automaticamente parte del codice a partire dai diagrammi ad oggetti.
Proprio gli anni Novanta vedono la progressiva diffusione degli ***strumenti di CASE*** che offrono un'ampia copertura del ciclo di vita delle applicazioni,fornendo assistenza nelle fasi di analisi,progettazione,produzione automatica del codice e testing.
Un punto di svolta nell'evoluzione degli strumenti e dei metodi di sviluppo è rappresentato dall'avvento dell'*Unified Modelling Language*(*UML*) [2],una notazione che racchiude concetti provenienti dalle metodologie ad oggetti , accettata oggi come standard *de facto* per la progettazione e largamente condivisa da analisti e sviluppatori.
Il capostipite dei CASE tools basati su UML,***Rational Rose***, è ormai ampiamente diffuso nelle aziendee consente la gestione collaborativa e la manutenzione di progetti di grandi dimensioni.

Le moderne ***Data-Intensive Web Applications*** differiscono in vari aspetti dalle applicazioni informatiche tradizionali:

- Sfruttano un'architettura client-server basata sul protocollo HTTP e su client leggeri("thin client"),spostando così gran parte delle funzioni applicative al lato server.
- Fanno uso di linguaggi di mark-up,come HTML, per la costruzione dell'interfaccia utente,il che comporta spesso la generazione dinamica delle interfacce a tempo di esecuzione.
- L'accesso universale da parte di utenti con poca abilità nell'uso di tali applicazioni web introduce la necessità di nuove interfacce in grado di catturare l'attenzione dell'utente e facilitare l'accesso all'informazione.Assumono dunque molta importanza aspetti comunicativi come la qualità della veste grafica,la personalizzazione , la coerenza ed usabilità dell'interfaccia utente e l'offerta all'utente dell'esplorazione libera dell'informazione mediante la navigazione ipertestuale.
- La disponibilità globale di sorgenti di informazione eterogenee richiede una gestione integrata di dati strutturati (ad esempio,database records) e non-strutturati (ad esempio, elementi multimediali), memorizzati in sistemi differenti(database, file systems, multimedia storage devices) e distribuiti su più siti.

Tali caratteristiche rendono lo sviluppo di queste applicazioni un'attività molto complessa che richiede tutti gli strumenti e le tecniche dell'ingegneria del software ,tra cui un processo di sviluppo del software ben organizzato, linee guida su come condurre le varie attività e ,sopratutto, appropriati concetti di design e notazioni.

Nasce dunque la necessità di poter applicare i benefici della modellazione concettuale riscontrati nelle applicazioni software tradizionali anche alle *Data-Intensive Web Applications* ,le quali devono essere specificate utilizzando una notazione grafica, intuitiva e di alto livello,facilmente comunicabile ad utenti non tecnici e di grande aiuto per l'implementazione dell'applicazione.
Le metodologie tradizionali risultano inadeguate per la modellazione della parte specifica delle *Web Applications* che riguarda l'interfaccia ipertestuale verso l'utente.Notazioni standard come UML descrivono un "mondo qualsiasi" in termini di oggetti ed associazioni tra oggetti. Nel contesto Web è necessario definire quali siano gli oggetti "giusti" per rappresentare gli elementi e meccanismi caratteristici del front-end delle applicazioni Web e costruire strumenti in grado id fornire un'automazione maggiore di quella offerta oggi dagli strumenti basati sulla modellazione ad oggetti "generale" di UML.

1.3 I componenti dello strato front-end di una web application

Nella fase di modellazione la scelta di quali elementi del sistema modellare è di fondamentale importanza in quanto permette di delineare la realtà che si vuole rappresentare stabilendo quali sono le caratteristiche significative del contesto applicativo da modellare.
L'architettura software di riferimento dell'applicazione è il soggetto della modellazione , infatti ad ogni livello architetturale si devono individuare gli elementi più significativi da modellare e stabilire il corretto livello di astrazione e di dettaglio da seguire nella costruzione del modello.
L'architettura di riferimento per le *Web Applications* è un'architettura

multilivello (*Multi-tier Architecture*) , specializzazione nel caso "web" della più generale architettura software a tre livelli (*Three-tier Architecture*).
Entrambe le architetture vengono descritte in seguito con particolare risalto all'individuazione dei componenti del *front-end tier* delle applicazioni web per le quali l'interfaccia verso l'utente e l'interazione con esso risultano di fondamentale importanza e quindi necessitano di essere modellate.

1.3.1 Architettura software a tre livelli

Nelle architetture a tre livelli ,a differenza di quelle a due livelli dove i programmi del client interagiscono direttamente con il DBMS ,è presente un livello intermedio tra il client ed i dati che concentra la logica di business dell'applicazione.
Tale livello,detto *Middle tier*,realizza la gestione dei processi per l'esecuzione della logica di business.
Disporre di un livello intermadio nel quale iene centralizzata la logica di processo rende più facile la gestione delle modifiche localizzando le funzionalità del sistema in modo che sia sufficente che i cambiamenti vengano scritti una sola volta e posti nel middle-tier perchè siano visibili da tutto il sistema.
L'architettura a tre livelli viene utilizzata quando è necessario un effettivo design client-sever distribuito in quanto aumenta le prestazioni,la flessibilità e la scalabilità mascherando all'utente la complessità dei processi distribuiti.

Nelle *Three-tier Architectures*(vedi figura 1.1)si distinguono i seguenti livelli:

- *Tier 3: Data tier*
  Tale livello riguarda l'organizzazione dei dati in databases o in file systems e la gestione di questi.La consistenza dei dati viene assicurata attraverso l'uso del data locking e della

replicazione.Questo livello è dedicato ai servizi legati ai dati o ai files che possono essere ottimizzati senza utilizzare il linguaggio specifico di un particolare DBMS.

- *Tier 2: Middle tier*
  Tale livello rappresenta la gestione della logica dell'applicazione: vengono create le corrispondenze tra i dati memorizzati nei files o databases e l'informazione accessibile dall'utente, in modo da permettere l'accesso a strutture dati complesse e realizzare le operazioni in tempi rapidi.In tale livello vengono condensate parti consistenti della logica di business (*business logic*) riguardanti la gestione dei processi relativi ai servizi che si devono fornire (*Middle-Tier Services*).

- *Tier 1: Client tier*
  Tale livello rappresenta l'interfaccia tra il sistema e l'utente finale; dunque la gestione di tutti i servizi legati all'interazione con l'utente,come la gestione della sessione di lavoro, del dialogo con l'utente e dei dati forniti in input dall'utente.

Le architetture a tre livelli facilitano lo sviluppo del software poichè ogni livello può essere costruito ed eseguito su una piattaforma separata e sviluppato in un linguaggio diverso,in modo che l'organizzazione dell'implementazione risulta più facile.

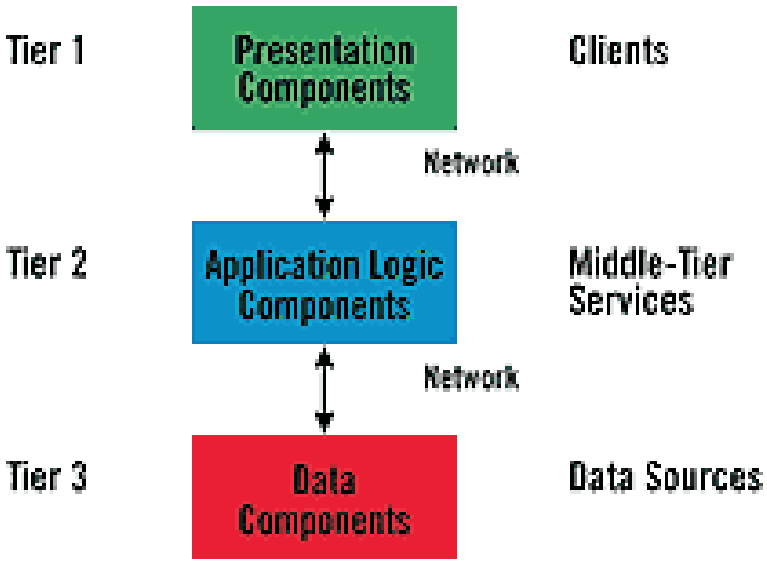

Figure 1.1: Generica architettura a tre livelli

1.3.2 Architettura multilivello di un'applicazione web

Spesso il livello intermedio delle architetture a tre livelli, il *Middle tier*,è diviso in due o più unità con differenti funzioni : tali architetture sono dette *Multi-tier Architectures*.È questo il caso delle attuali applicazioni web su larga scala, le quali presentano un'architettura software di riferimento a più livelli (vedi figura1.2).
In tale architettura il livello logico intermedio, il *Middle tier* risulta suddiviso in : *Presentation layer* e *Business Logic layer*. Il *Presentation layer* genera le pagine web ed il loro contenuto dinamico che generalmente proviene da un database (ad esempio una lista di prodotti con determinate caratteristiche,una lista delle transazioni effettuate nell'ultimo mese,etc.).L'altro compito fondamentale del *Presentation layer* è quello di "decodificare" le informazioni ritornate dal client,cioè individuare i dati forniti in input dall'utente ed inviare tale informazione al *Business Logic layer*.Il livello logico *Presentation layer* viene imple-

mentato all'interno di un *Web Server*.
Il *Business Logic layer* rappresenta la logica dell'applicazione, le funzionalità di tale livello sono:realizzare tutti i calcoli e le validazioni richieste,gestire il "workflow",cioè il flusso di lavoro che gestisce gli eventi provocati dall'utente(incluso tenere traccia della data della sessione) e l'accesso ai dati per il *Presentation layer*.Tale livello logico viene implementato all'interno di un *Application Server*.Un *Application Server* è dunque una piattaforma software , distinta dal web server,dedicata ad un'esecuzione efficente del business per supportare la costruzione di pagine dinamiche.
Nelle architetture moderne generalmente il *Web Server*risulta molto semplificato e si limita alla gestione delle richieste HTTP del client,spostando tutta la logica di generazione delle pagine nell'*Application Server*.

Individuazione dei componenti front-end

Il **front-end** di un'applicazione ,indipendentemente dalla distribuzione delle funzionalità tra i dispositivi ai vari livelli dell'architettura di riferimento,è composto dagli elementi e meccanismi tecnologici che costituiscono l'interfaccia dell'applicazione verso l'utente finale in contrapposizione al **back-end**,il quale fa riferimento alla gestione ed al mantenimento dei dati di interesse.

Le applicazioni web presentano un **front-end ipertestuale** per la visualizzazione dell'informazione e la realizzazione di servizi.
Un *ipertesto*è un insieme di nodi connessi tramite dei links,dove ogni nodo rappresenta un concetto ed i links collegano concetti in relazione tra loro.
Analizzare e modellare l'*ipertesto* significa dunque specificare l'organizzazione dell'interfaccia di front-end di un'applicazione web.

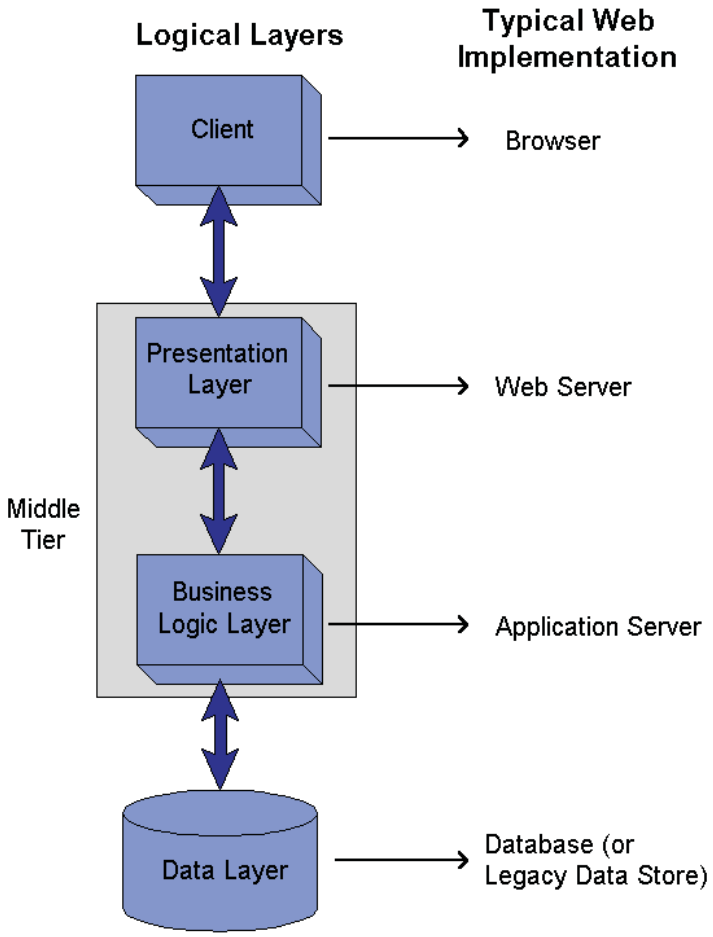

Figure 1.2: Architettura multilivello di un'applicazione web

Gli aspetti che compongono tale front-end ipertestuale sono gli elementi caratteristici delle applicazioni web:la divisione logica dell'applicazione in moduli e la topologia dell'ipertesto di ogni modulo in termini di pagine,che contengono dati ed informazioni e sono collegate tra loro per poter supportare la navigazione dell'utente e l'interazione con esso.Gli elementi del front-end da modellare sono dunque le *pagine*,i *links* e la generazione dinamica del contenuto delle pagine sulla base delle azioni dell'utente.

Se si tiene conto della distribuzione delle funzionalità tra i dispositivi ai vari livelli logici dell'architettura web e quindi si analizzano l'*ipertesto* ed i suoi componenti ad un minore livello d'astrazione rispetto all'organizzazione delle pagine e dei links, per ogni pagina si possono considerare l'aspetto client e l'aspetto server e quindi individuare meccanismi tecnologici ,sia server-side che client-side, legati al front-end ipertestuale.

Dal lato client,ogni pagina web richiesta dal browser al web server è un insieme di contenuti e di istruzioni di formattazione,espresse in HTML.Le pagine web possono includere *client-side scripts* che definiscono un ulteriore comportamento dinamico per la visualizzazione delle pagine ed interagiscono con il contenuto delle pagine e con altri controlli (*Applets* e *ActiveX controls*)all'interno delle pagine stesse.Dalla prospettiva del client dunque una pagina web è un documento HTML formattato e gli elementi da considerare nella modellazione del front-end ipertestuale sono i *client-side scripts* (come *JavaScript*), le *Applets* e gli *ActiveX controls*.

Dal lato server,la visione delle pagine web risulta più complicata.Nelle prime applicazioni web le pagine dinamiche erano costruite attraverso la *Common Gateway Interface*(*CGI*),la quale definiva un'interfaccia standard per i programmi allo scopo di accedere all'informazione richiesta.Ogni volta che la richiesta includeva una URL riferita ad un programma eseguibile, il *CGI script*,il web server eseguiva tale programma ritornando all'utente l'output dell'esecuzione.Oggi i web server sono stati estesi con *application execution engine*,ambienti di esecuzione efficente e persistente dove vengono processate le pagine web (esempi sono le Microsoft's Active Pages ASP e le Java Server Pages JSP).
Dalla prospettiva del server dunque elementi significativi sono i

server-side scripts e le risorse server-side come i componenti di accesso ai database .
Altri meccanismi da considerare sono :

- Le *Forms* , il principale meccanismo di data entry per le pagine web legato sia al client che ne deve compilare i campi che al server che ne deve processare l'invio.
- I *Frames* , i quali permettono che più pagine siano attive e visibili all'utente e dunque rappresentano un comportamento legato all'aspetto client.

Gli elementi ,come le pagine ed i links, e i meccanismi tecnologici individuati sono componenti specifici delle applicazioni web e del loro front-end ipertestuale; si ha la necessità di modellare la complessità della loro interazione e realizzare la loro integrazione con i modelli del resto del sistema .

Chapter 2

Requisiti necessari per un linguaggio di modellazione web

2.1 Categorizzazione dei requisiti sulla base dell'architettura informativa e funzionale

Per uno sviluppo efficace delle applicazioni web si devono considerare due aspetti dell'architettura tecnica:l'architettura dell'informazione e l'architettura funzionale [3].
I linguaggi di modellazione web,usati nello sviluppo di tali applicazioni, devono essere in grado di catturare e comunicare entrambi gli aspetti.In particolare,un linguaggio di modellazione web(WML[1]) deve mettere a disposizione funzioni per facilitare la comprensione, specifica, comunicazione, visualizzazione e costruzione di un design dettagliato.
Si possono dunque individuare due gruppi di requisiti necessari per un WML relativi agli aspetti dell'informazione e della funzionalità,insieme ad alcuni requisiti più generali.

[1]Usiamo WML per indicare un Web Modelling Language

2.1.1 Requisiti funzionali

Le caratteristiche funzionali delle applicazioni web impongono alcuni requisiti ai linguaggi di modellazione utilizzati per rappresentare tale funzionalità.
Molti di questi requisiti sono relativi alla natura specifica dell'architettura di tali applicazioni.

Capacità di modellare integrazione e connettività

In molte organizzazioni ,le nuove web applications lavorano a stretto contatto con sistemi legacy preesistenti tipicamente orientati verso il back-end.La necessità dell'integrazione dei componenti web introduce requisiti specifici per i WMLs.Infatti,poichè l'integrazione di componenti dipende in gran parte dalle definizioni dell'interfaccia,i WMLs devono essere in grado di modellare e documentare le interfacce dei componenti in modo accurato e privo di ambiguità.
Come interfaccia di front-end di molte organizzazioni, le applicazioni web connettono una grande varietà di sorgenti di informazione e di servizi;dunque i linguaggi di modellazione devono supportare la rappresentazione della connettività e dei meccanismi di accesso.

Abilità di supportare l'uso di patterns

L'esperienza sia pratica che di ricerca suggerisce che è auspicabile utilizzare dei patterns nella modellazione di sistemi software.I benefici potenziali dei patterns includono:

- L'uso di patterns fornisce un'aiuto per lo sviluppo del software,in modo che le decisioni per un design corretto possano essere prese con il minimo sforzo.

- Poichè molti patterns appropriati sono comunemente accettati come standard di alta qualità, il loro utilizzo può aumentare la qualità totale dell'applicazione web.

- L'uso di patterns può aumentare l'interconnettività dell'applicazione, in quanto tali patterns sono normalmente condivisi dai vari sviluppatori software,specialmente in domini applicativi simili.

Abilità di rappresentare i concetti indipendentemente dalla tecnologia

Poichè le tecnologie nell'ambito del web cambiano molto rapidamente è impossibile e poco utile effettuare un design delle applicazioni web con riferimento ad una specifica tecnologia.
È dunque necessario che i WMLs permettano di rappresentare i concetti specifici del web indipendentemente dalla tecnologia.

2.1.2 Requisiti legati all'informazione

I requisiti legati all'informazione tipicamente coprono aspetti come:la gestione del contenuto,come viene strutturata l'informazione e l'accesso ad essa,la contestualizzazione dell'utente ,il design della navigazione, i punti di vista dell'informazione ed i problemi di presentazione.

Abilità di modellare concetti di presentazione

Il design delle applicazioni web legato alla presentazione ha delle caratteristiche specifiche, tra cui:

- Funzioni più complete e sofisticate a livello di presentazione,come conseguenza della necessità di aumentare la qualità delle interfacce con l'utente.

- Varie tipologie di dati a livello di presentazione,come testi,grafici,video ed audio.

- I WMLs devono essere usati da personale tecnico ,ma anche da designers grafici,autori,analisi di mercato in quanto lo sviluppo di applicazioni web è un'attività che racchiude non solo elementi tecnici ma anche manageriali, sociali ed artistici.

I WMLs devono dunque supportare la modellazione in un modo che sia consistente con tali caratteristiche.

Abilità di modellare la navigazione

È necessario modellare l'aspetto strutturale e comportamentale della navigazione, come uno degli elementi più caratteristici delle applicazioni web.Dal punto di vista strutturale si devono modellare i links tra le pagine web, i quali possono essere:

- *Contestuali*: connettono pagine in modo coerente con le relazioni semantiche tra i concetti che tali pagine contengono.Un link contestuale permette il passaggio di informazione(chiamata *contesto* tra le pagine.

- *Non-Contestuali*: connettono le pagine in modo totalmente libero,cio è indipendentemente dai loro contenuti e dalle relazioni semantiche tra questi.

Dal punto di vista comportamentale si tratta di modellare i comportamenti legati alla navigazione,ad esempio la decisione a runtime dell'endpoint di un certo link.

Abilità di modellare le interazioni tra utente ed informazione

Rispetto ai sistemi software tradizionali,i sistemi web tendono ad avere interazioni più sofisticate con gli utenti.Di conseguenza i WMLs devono essere in grado di modellare queste particolari interazioni in modo completo e non ambiguo affinchè possano essere individuate,comprese,modellate ed implementate.Nel rapporto con l'utente si possono individuare alcune caratteristiche rilevanti,discusse di seguito.

Le differenti sorgenti di informazione
Le applicazioni web devono connettere varie sorgenti di informazione differenti tra loro come,ma non solo, database,file server o document repositories.I WMLs devono modellare le differenti sorgenti di informazione ed i meccanismi di comunicazione ad esse associati.

Il modo d'accesso
Le applicazioni web possono interagire con le sorgenti di informazione in *modo reattivo* o in *modo proattivo*.
Nel modo reattivo,l'azione d'accesso è richiesta dall'utente e viene eseguita dall'applicazione web;mentre nel modo proattivo l'azione di accesso viene iniziata automaticamente dall'applicazione stessa.
I WMLs devono permettere di poter differenziare e modellare questi due modi d'accesso.

La distribuzione dell'informazione
La distribuzione dell'informazione deve essere supportata dall'architettura tecnica con una connettività costante ed affidabile,richiede il supporto della definizione del profilo utente in modo che l 'informazione venga fornita all'utente o gruppo di utenti interessato ed inoltre richiede un controllo della sicurezza per assicurare

la segretezza dell'informazione.I WMLs devono supportare tutti questi aspetti e specialmente la loro connessione.

Abilità di modellare i ruoli di utenti e gruppi di utenti

Il successo delle applicazioni web dipende in gran parte dalla soddisfazione dell'utente che si può ottenere ,ad esempio,fornendo funzionalità sofisticate, semplici interfacce ed una navigazione ben strutturata.Lo scopo è quello di evitare il disorientamento dell'utente fornendo un conteso adatto e funzioni di orientamento in grado di fornire all'utente la chiara percezione della propria posizione nell'"information space".
WMLs devono dunque presentare tali capacità,incluse la definizione dei profili utente edei gruppi di utenti, la connessione tra utenti e gruppi di utenti ed il legame tra profili utente e le interazioni degli utenti stessi.

Abilità di modellare il contenuto

La modellazione della struttura del contenuto informativo,spesso espressa come lo schema di un database, risulta di fondamentale importanza per le applicazioni web che spesso devono gestire grandi quantità di dati.I WMLs devono dunque fornire la possibilità di modellare il contenuto informativo proponendo nuove notazioni o utilizzando le tecniche tradizionali (come il modello E/R o il diagramma delle classi UML).

2.1.3 Requisiti generali

Abilità di collegare funzionalità ed informazione

L'integrità e la coesione di un'applicazione web dipende da una stretta e flessibile connessione tra i due aspetti dell'architettura tecnica analizzati:quello funzionale (sez. 2.1.1) e quello legato all'informazione (sez. 2.1.2).
I WMLs devono dunque supportare non solo la modellazione di elementi funzionali ed informativi,ma anche l'integrazione di essi in modo coesivo e consistente.

Abilità di mantenere l'integrità del sistema

L'integrità di un sistema web può essere compromessa da:

- La complessità delle applicazioni web su larga scala,ricche di funzioni complesse;

- I frequenti cambiamenti deirequisiti da parte di clienti incerti;

- I rapidi cambiamenti delle tecnologie;

- La combinazione di varie discipline coinvolte nello sviluppo delle applicazioni web.

I WMLs possono mantenere tale integrità fornendo definizioni semantiche sia per gli elementi dei modelli che per le relazioni tra di loro.Con l'aiuto di CASE tools l'integrità ed il controllo referenziale possono essere realizzati in modo automatico o semi-automatico.

Abilità di modellare a vari livelli di astrazione

Nelle varie fasi del ciclo di vita di un'applicazione web,come di un prodotto software in generale, i modelli vengono utilizzati come supporto (sez 1.1).Per soddisfare i requisiti specifici di ogni fase, i WMLs devono fornire modelli a diversi livelli di astrazione (sez 1.2) ed inoltre devono supportare la coesione dell'intero modello mettendo a disposizione interconnessioni coordinate tra questi livelli di astrazione.

Abilità di supportare la gestione del ciclo di vita delle applicazioni web

Per la maggioranza delle applicazioni,tra cui le applicazioni web, la fase di progettazione è solo l'inizio dell'intero ciclo di vita.Nelle applicazioni web in particolare, la fase di mantenimento assume un ruolo sempre più importante.I WMLs devono dunque facilitare la gestione di tutto il ciclo di vita delle applicazioni web,non solo della fase di progettazione.

Supporto di un CASE tool

Come sottolineato nella sez 1.1, per un linguaggio di modellazione è di fondamentale importanza fornire notazioni rappresentabili dagli strumenti di design.In particolare un WML deve essere facilmente supportato dalle tool suites commerciali conosciute dagli sviluppatori,come Entity-Relationship e UML editors e code generators,oppure da un CASE tool specifico per quel linguaggio.

2.2 Categorizzazione dei requisiti sulla base di tre dimensioni ortogonali

I requisiti necessari per i linguaggi di modellazione web possono essere categorizzati anche sulla base delle tre dimensioni ortogonali da considerare nella modellazione delle applicazioni web ,come mostra la figura 2.1:*livelli*,*aspetti*e *fasi* [4].

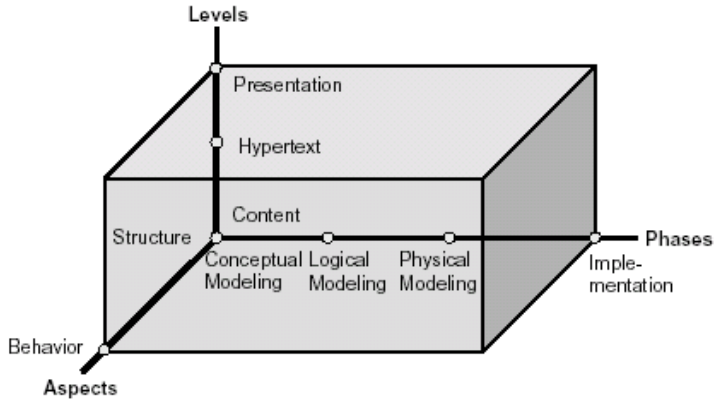

Figure 2.1: Dimensioni della modellazione di applicazioni web

2.2.1 Livelli:Contenuto,Ipertesto,Presentazione

La prima dimensione nella modellazione delle applicazioni web comprende tre livelli differenti:

- *Livello di contenuto*: fa riferimento ai dati usati dall'applicazione ed è spesso gestito tramite database systems;
- *Livello di ipertesto*:denota la composizione logica delle pagine e la struttura della navigazione;
- *Livello di presentazione*:riguarda la rappresentazione grafica del livello di ipertesto,cio è il layout di ogni pagina e l'interazione con l'utente.

La separazione tra i livelli ed il mapping esplicito : è necessario che vi sia una chiara separazione tra questi tre livelli,ogniuno dei quali riguarda un particolare aspetto delle applicazioni web.Tale separazione può essere ottenuta rendendo esplicite le interdipendenze,cioè il mapping, tra i livelli,facilita l'evoluzione del modello,riduce la complessità ed aumenta la flessibilità.Deve essere mantenuta:

- L' *indipendenza tra dati e presentazione*:richiede che la descrizione logica dei dati sia indipendente dal modo in cui i dati vengono presentati all'utente,ciò permette di cambiare il formato della presentazionesenza alterare lo schema del database;

- L' *indipendenza tra dati ed ipertesto*:favorisce l'uso dei dati da parte di applicazioni differenti e fornisce la possibilità di avere diverse viste dell'informazione sulla base del profilo dell'utente;

- L' *indipendenza tra ipertesto e presentazione*:permette di avere presentazioni diverse per lo stesso ipertesto sulla base delle specifiche del browser o di elementi di personalizzazione,aumenta la portabilità rendendo possibile generare appliczioni su piattaforme diverse.

Possibilità di mapping flessibili: Le possibilità di mapping tra i vari livelli devono essere il più flessibili possibile e devono assicurare che si ottenga la derivazione di un livello dall'altro ,nonostante le differenze tra i vari livelli(ad esempio,la ridondanza dei dati è presente a livello di presentazione per facilitare la navigazione ma è eliminata a livello di contenuto con le tecniche di normalizzazione).

Bottom-up e Top-down design: Un'altro requisito riguardante questi livelli è che deve essere possibile seguire non solo un ap-

proccio bottom-up nel design(cominciare a modellare il livello di contenuto e poi derivare i modelli degli altri livelli),ma anche un design top-down in cui il livello di contenuto viene derivato dagli altri livelli.

2.2.2 Aspetti:Struttura e Comportamento

La seconda dimensione comprende gli aspetti di *struttura* e *comportamento* ,i quali sono ortogonali rispetto ai tre livelli della prima dimensione,cio è ad ogni livello devono essere modellati sia gli aspetti strutturali che quelli comportamentali.

1. A livello di contenuto:

   Struttura : riguarda la creazione della struttura del dominio mediante i meccanismi di astrazione standard come classificazione, aggregazione e generalizzazione.

   Comportamento : l'aspetto comportamentale riguarda l'application logic dipendente dal dominio.

2. A livello di ipertesto:

   Struttura : riguarda la composizione delle pagine e le relazioni tra esse in termini di navigazione.

   Comportamento : un esempio di aspetto comportamentale legato all'ipertesto è la la decisione a runtime dell'endpoint di un certo link.

3. A livello di presentazione:

   Struttura : riguarda gli elementi dell'interfaccia con l'utente e la loro composizione gerarchica.

   Comportamento : riguarda le reazioni agli eventi di input (le azioni dell'utente,come premere un bottone),l'interazione

e la sincronizzazione tra gli elementi dell'interfaccia con l'utente.

Anche se si potrebbe scegliere un diverso formalismo per ogni livello,sarebbe molto utile l'uso di un unico formalismo per rappresentare struttura e comportamento a tutti i livelli;anche allo scopo di rendere più semplice il mapping tra i vari livelli.
Un altro requisito per facilitare la rappresentazione dei due aspetti indicati è che i WMLs devono supportare l'uso di patterns per il design a tutti i livelli.La maggior parte dei patterns disponibili riguarda i livelli di ipertesto e presentazione.

2.2.3 Fasi:Analisi,modellazione concettuale,logica,fisica ed implementazione

La terza dimensione per la modellazione delle applicazioni web comprende le fasi del ciclo di vita di un prodotto software,dall'*analisi* all'*implementazione*.
Questa dimensione è ortogonale alle due presentate precedentemente,infatti la struttura ed il comportamento del contenuto,dell'ipertesto e della presentazione devono essere modellati in ogni fase del processo di sviluppo.Non c'è un consenso generale sul modello per il ciclo di vita delle applicazioni web,tuttavia si può considerare appropriata una distinzione,come nel design dei databases, tra una rappresentazione astratta del dominio chiamata *modellazione concettuale*,un design indipendente dalla tecnologia,*modellazione logica* ed un design dipendente dalla tecnologia,*modellazione fisica*(sez. 1.2).

Chapter 3

Analisi e confronto dei linguaggi proposti per la modellazione di interfacce web

Il capitolo si propone di valutare lo stato dell'arte dei linguaggi per la modellazione di interfacce web.
Tali formalismi vengono analizzati sulla base del linguaggio di modellazione tradizionale da cui derivano e delle interdipendenze tra essi e confrontati tra loro sulla base dei requisiti individuati nel capitolo 2.
I linguaggi e formalismi per la modellazione di interfacce web hanno origine da tre principali ambiti del software:

- *Hypermedia*,avente come base il *Dexter Reference Model* [5]:
  - *HDM* [6][7]
  - *HDM-lite/AutoWeb*[8]
  - *WebML* [9] [10]
- *Database systems*,basati sul modello *Entity-Relationship*(E/R)[1]:
  - *RMM*[11]
  - *Araneus*[12]
  - *Strudel* [13]

- *Object-oriented modelling*,in termini di *Object Oriented Technique*(*OMT*) [14] e di *Unified Modelling Language*(*UML*)[2]:
  - *OOHDM*[15] [16] [17]
  - *Metodo OO-H*[18]
  - *UML esteso di Conallen* [19] [20]
  - *Koch-UWE*[21] [22] [23]

La figura 3.1 mostra le origini differenti dei vari linguaggi di modellazione,le frecce rappresentano le interdipendenze tra tali linguaggi proposti. Coerentemente con tali dipendenze,i linguaggi di modellazione vengono classificati in 3 generazioni:dai formalismi di prima generazione a quelli di seconda generazione fino alle proposte più recenti.
Se si escludono approcci indipendenti ,come Conallen e Strudel che derivano direttamente da formalismi di base come UML e E/R rispettivamente,la maggior parte dei linguaggi di ultima generazione citati si basano su metodi di modellazione precedenti aventi origini in ambiti del software differenti rispetto alle tecniche che vengono effettivamente utilizzate per la modellazione.Ad esempio,come mostra la figura 3.1,i linguaggi RMM e OOHDM derivano entrambi dall'ambito dell'*Hypermedia* e da HDM ma applicano tecniche differenti:le tecniche dell' *Entity-Relationship* e dell'*Object Oriented* rispettivamente.
È importante sottolineare che ,data la rapida evoluzione del Web,i formalismi delle tre generazioni sono stati concepiti per far fronte ad esigenze di modellazione molto differenti.
Le metodologie delle prime generazioni sono nate per modellare siti web del tipo "sola lettura" e dunque rappresentano i primi tentativi di modellare elementi ipermediali come le pagine,i links e la navigazione.Tali metodologie si concentrano sull'organizzazione delle strutture dell'informazione e dei cammini di navigazione e ,per lo più,si sono mantenute ad un livello propositivo teorico.

Le proposte più recenti devono invece far fronte alle esigenze di modellazione delle moderne applicazioni web che comportano la gestione di grandi quantità di dati e di operazioni per la modifica di tali dati che risultano sempre più complesse e vanno ben oltre la semplice operazione di "seguire un link".I formalismi più recenti propongono notazioni per modellare tali aspetti dinamici delle moderne applicazioni web e sono inseriti all'interno di processi completi di sviluppo di applicazioni web supportati da CASE tools per la generazione automatica o semi-automatica del codice.
Naturalmente ,allo scopo di valutare lo stato dell'arte dei linguaggi di modellazione web,si devono analizzare e confrontare le proposte di ultima generazione;tuttavia ,sulla base delle interrelazioni tra i linguaggi rappresentate in figura 3.1, si ritiene sia utile descrivere e comprendere anche le proposte meno recenti dalle quali derivano i progetti di ultima generazione.

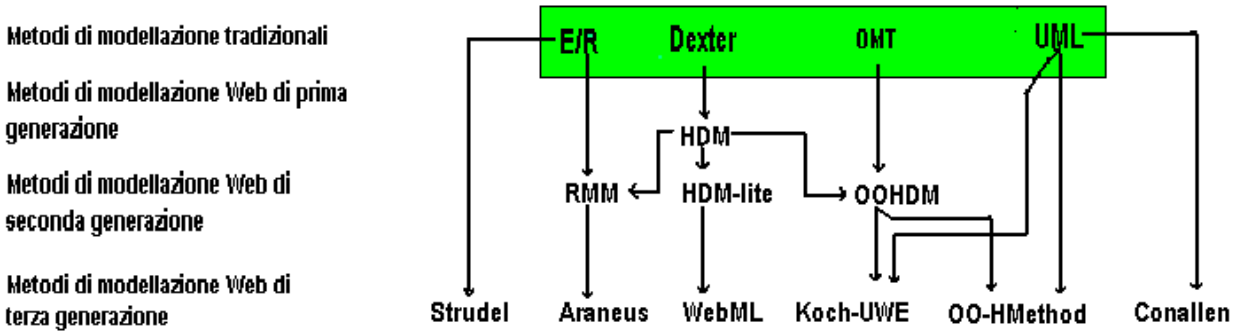

Figure 3.1: Origini e relazioni dei linguaggi di modellazione Web

3.1 Linguaggi di modellazione web nell'ambito dell'*Hypermedia*

Prima di analizzare i linguaggi di modellazione aventi origine in tale ambito,è utile fornire una breve definizione di *Ipertesto*,*Presentazione multimediale* ed *Hypermedia*.

- Un *Ipertesto* è un insieme di concetti e links che collegano tra loro tali concetti;viene modellato come una rete di nodi collegati da un insieme di links;
- Una *Presentazione multimediale* è caratterizzata da una combinazione di testi,suoni,video e sopratutto si differenzia da altre presentazioni per l'interazione dell'utente;
- Un *Hypermedia* è una combinazione di ipertesto e presentazione multimediale:i links possono essere definiti tra un

qualunque insieme di oggetti multimediali,inclusi suoni,video e realtà virtuale.

Il primo modello nell'ambito dell'*Hypermedia* ad essere accettato fu il *Dexter Reference Model* [5].
Tale modello forniva una terminologia uniforme per rappresentare le diverse primitive messe a disposizione dai sistemi per la costruzione dell'ipertesto.Nel *Dexter Reference Model* i componenti rappresentano le parti di informazione che costituiscono l'ipertesto ed i links rappresentano la navigazione,ossia i cammini percorribili.
In seguito molte proposte nel campo dell'*Hypermedia* si sono basate sul *Dexter Reference Model* ed hanno aggiunto primitive di modellazione pi sofisticate,semantiche formali e processi di sviluppo strutturati.

3.1.1 HDM:Hypertext Design Model

L'*Hypertext Design model*(*HDM*)[6] è stato uno dei primi metodi sviluppati per definire la struttura e l'interazione nell'ambito dell'*Hypermedia*.Come mostra la figura (figura3.1),HDM è stato sviluppato a partire dal *Dexter Reference Model* ma si basa sulla metodologia *Entity-Relationship*(*E/R*),che viene estesa con un'organizzazione gerarchica.
HDM introduce un chiara separazione tra due aspetti:

- *Authoring-in-the-large*:comprende la modellazione dell'applicazione da un punto di vista strutturale e globale,individuando elementi generali come entità,componenti o unità e collegando caratteristiche comuni ad applicazioni di un certo dominio.

- *Authoring-in-the-small*:fa riferimento a dettagli dell'organizzazione e del comportamento di una specifica applicazione e dunque dipende dagli strumenti di implementazione.

HDM permette di identificare non solo i componenti atomici dell'ipertesto ma anche i criteri per assemblare le strutture.HDM in verità tratta solamente gli aspetti dell'*Authoring-in-the-large* e descrive solo la struttura dell'ipertesto ,senza considerare specifici dettagli implementativi.
Tale struttura viene rappresentata estendendo il concetto di *entià* del modello E/R ed introducendo nuove primitive come le *unità*(corrispondenti ai "nodi" dell'ipertesto) ed i *links*.
Un'*entità* è la più piccola parte di informazione che può essere aggiunta o cancellata da un'applicazione;per esempio sono entità un'opera lirica("Le Quattro Stagioni" di Vivaldi,ad esempio) o un pittore ("Piero della Francesca"ad esempio) o una legge (ad esempio "legge 8/11/2000").

Le *entità HDM* rappresentano oggetti fisici o concettuali del dominio; si differenziano da quelle del modello E/R in quanto hanno una loro struttura interna gerarchica ed hanno associata una semantica per il browsing,cioè la specifica di come può essere realizzata la navigazione e come viene visualizzata l'informazione.
Ogni entità è una gerarchia di *componenti* a loro volta costituiti da *unità*.
I *componenti* non sono autonomi ed esistono solo come parte di un'entità;ad esempio possiamo considerare "Primavera" un componente dell'entità "Le Quattro Stagioni" o "Il Battesimo di Cristo"un componente dell'entità "Piero della Francesca".
Ogni *unità* che costituisce un componente mostra il contenuto del componente da una particolare *prospettiva*.
Una *prospettiva* è una presentazione differente dello stesso contenuto;ad esempio in un'applicazione multilingua lo stesso argomento pu ò essere descritto con linguaggi diversi.

Un'*unità* è caratterizzata da un *nome* e da un *body*.Inserire i contenuti nei *bodies* appropriati rappresenta l'*Authoring-in-the-*

*small.*Ad esempio "Piero della Francesca/vita italiana" è l'unità che descrive in italiano la vita del pittore.Tutti i componenti di un'entità devono avere associato lo stesso numero di prospettive dell'entità stessa.
Nella tabella3.1,ad esempio, ogni componente dell'entità "Le Quattro Stagioni" contiene una registrazione digitale,un punteggio musicale,commenti al testo in inglese ed in italiano.

ENTITÀ		"Le Quattro Stagioni" di Vivaldi		
COMPONENTI	primavera	estate	inverno	autunno
PROSPETTIVE	Registrazione Digitale UNIT	Punteggio Musicale UNIT	Commenti Inglese UNIT	Commenti Italiano UNIT

Table 3.1: Esempio di composizione di un'entità HDM

Estendendo la definizione usata nell'ambito dei database,uno *schema HDM* è un insieme di definizioni di entità e links,mentre un'*istanza* dello schema è un insieme delle entità e dei links effettivi che rispettano i vincoli definiti nello schema.
L'ipertesto di una applicazione è composto da un'*iperbase*,formata da entità,componenti,unità e links,e da un insieme di strutture d'accesso,dette *outlines.*Un *outline* è una parte dell'ipertesto contenente informazioni sulla navigazione che permette l'accesso alle entità dell'*iperbase* offrendo un entry point per cominciare la navigazione e la possibilità di individuare e selezionare le entità.
Gli outlines definiti in HDM sono:

- *Index links*:connettono il nodo rappresentante una collezione con ogni membro della collezione.

- *Guided tours*:connettono tutti i nodi rappresentanti le collezioni in una sequenza lineare in cui ogni membro è collegato con

il precedente e con il successivo. Nelle collezioni circolari l'ultimo membro è collegato con il primo.

- *Collection links*: sono index o guided tour links che permettono di visitare i nodi della collezione.

Gli outlines permettono all'autore di organizzare un insieme di links nell'iperbase e non sono specificati nello schema,ma possono essere aggiunti nell'applicazione per fornire un'entry point all'utente.

Uno degli elementi di HDM che furono considerati più interessanti è l'introduzione di differenti categorie di links che tengono conto delle diverse relazioni esistenti tra gli elementi dell'informazione,come le relazioni tra entità e le regole di navigazione(come l'utente può muoversi all'interno della struttura ipermediale. In HDM vengono definiti tre tipi di link:

- *Links strutturali*:connettono i componenti.

- *Links di prospettiva*:permettono di collegare le unità con i componenti a cui fanno riferimento.

- *Links di applicazione*:connettono componenti ed entità dello stesso tipo o di tipo diverso e ,a differenza degli altri due tipi di link che possono essere derivati automaticamente dalla struttura delle entità,sono definiti dall'autore.

Per supportare il design della presentazione,cioè il design di come il contenuto e le funzioni dell'applicazione vengono presentate all'utente,HDM definisce due concetti:*slot* e *frame*.
Uno *slot* è una parte atomica dell'informazione ,può essere semplice o complesso,come un video sincronizzato con suoni.Un *frame* è un'unità di presentazione,cioè ciò che viene mostrato all'utente.

Analisi di HDM sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali di HDM descritti precedentemente permettono di verificare se l'Hypermedia Design Model soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,HDM distingue solo tra livello di ipertesto e livello di presentazione e,anche se la separazione tra tali livelli è chiara,non vengono descritti concetti per un mapping esplicito e flessibile tra i livelli.

- Dal punto di vista dei *aspetti*,in HDM gli aspetti strutturali sono considerati ad entrambi i livelli mediante l'uso dei concetti descritti precedentemente,mentre gli aspetti di comportamento vengono considerati principalmente a livello di presentazione modellando l'interazione con l'utente.

- Dal punto di vista delle *fasi*,HDM,come indicato precedentemente distingue tra le due fasi di *Authoring-in-the-large*e *Authoring-in-the-small*.

3.1.2 HDM-lite/AutoWeb

AutoWeb[8] è un progetto di ricerca che propone una metodologia ed uno strumento per lo sviluppo di siti Web Data-intensive. Il processo di sviluppo di un sito Web in AutoWeb è organizzato in tre fasi principali:

1. Definizione del sito web;

2. Generazione del database di supporto;

3. Implementazione del sito Web.

La prima fase è basata su uno spcifico linguaggio ,chiamato *HDM-lite*,e guida verso la definizione di un *iperschema*,cioè una descrizione ad alto livello degli elementi più rilevanti del sito indipendente dall'implementazione.
HDM-lite rappresenta un'evoluzione di HDM di cui condensa i concetti.
Un *iperschema* consiste di tre parti:

1. *Schema strutturale*:descrive le proprietà strutturali dei principali oggetti che compongono l'applicazione.

2. *Schema navigazionale*:viene imposto sullo schema strutturale allo scopo di specificare come sia possibile muoversi da un oggetto all'altro ed indicare i cammini di accesso per raggiungere gli oggetti.

3. *Schema di presentazione*:descrive come rappresentare graficamente gli schemi precedenti(strutturale e navigazionale).

Hdm-lite mette a disposizione un ricco insieme di costrutti di modellazione per descrivere le tre parti dell'iperschema.
Lo schema strutturale viene descritto da una variante del modello Entity-Relationship:oggetti della stessa classe sono descritti da *tipi di entità*,i quali possono aggruppare una gerarchia di *componenti*,cioè aggregazioni di pezzi atomici di informazione detti *slot*;inoltre vengono utilizzati *tipi di link* per descrivere le relazioni binarie tra i tipi di entità.
Per descrivere lo schema navigazionale HDM-lite mette a disposizione *traversals* e *collections*.
I *traversals* possono essere definiti sulla base dei *tipi di link*dello schema strutturale e vengono usati per specificare come muoversi da un oggetto ad un altro oggetto connesso.
Le *collections* vengono introdotte per decrivere la struttura di accesso.Sia i *traversals* che le *collections* sono associati con le semantiche orperazionali della navigazione,cioè con i dettagli che

specificano l'esecuzione a run-time di una navigazione.
Infine la presentazione è modellata in HDM-lite attraverso il *tipo di pagina*.Vi sono tre categorie di *tipi di pagina*:

- *Tipi di pagina component*:specificano la presentazione dei componenti;
- *Tipi di pagina collection*:specificano la presentazione delle collections;
- *Tipi di pagina traversal*:specificano la presentazione dei traversals;

I *tipi di pagina* vengono considerati come griglie astratte ed ogni elemento della griglia ha un'insieme di proprietà di presentazione che possono essere settate dall'utente.
Un ricco ambiente grafico assiste il designer in ogni passo del processo di definizione dell'iperschema.
Una volta terminata la definizione dell'iperschema,il sistema genera automaticamente il sito attraverso due passi successivi:inizialmente viene generato lo schema di un database relazionale a partire dallo schema strutturale;in seguito,sulla base del database,vengono generate le pagine HTML per lo schema navigazionale e per i *tipi di pagina*.

Analisi di HDM-lite/AutoWeb sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali di HDM-lite/AutoWeb descritti precedentemente permettono di verificare se tale progetto soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,come HDM il livello di contenuto non viene modellato separatamente ma piuttosto

insieme al livello di ipertesto attraverso lo schema strutturale,come descritto in precedenza.Inoltre,a livello di ipertesto, viene definito anche uno schema navigazionale.Il livello di presentazione viene modellato attraverso uno schema di presentazione.Manca dunque la separazione del livello di contenuto dagli altri due livelli;inoltre ,bench è sia possibile mappare pi ù schemi di presentazione sugli schemi strutturale e navigazionale,non vengono forniti costrutti espliciti per il mapping.

- Dal punto di vista dei *aspetti*,gli aspetti comportamentali non vengono trattati a nessun livello.

- Dal punto di vista delle *fasi*,HDM-lite propone una trasformazione degli schemi concettuali in una rappresentazione logica e in un ulteriore rappresentazione fisica.Per la prima parte,cioè la trasformazione da modellazione concettuale a modellazione logica,le tecniche ben conosciute di trasformazione dello schema concettuale E/R nello schema logico vengono usate per trattare anche gli aspetti navigazionali e di presentazione.Tali trasformazioni sono implementate dal sistema AutoWeb

3.1.3 WebML

WebML (Web Modelling Language)[9] [10] è un linguaggio proposto in sede internazionale da un gruppo di ricercatori di Milano che ,come suggerisce il nome,si propone di adattare le tecniche della modellazione concettuale alle caratteristiche distintive delle applicazioni web.
WebML può essere considerato il diretto discendente di AutoWeb,infatti WebML eredita da AutoWeb l'approccio basato sui modelli ,ma introduce meccanismi per la gestione di aspetti comportamentali e perciò supporta lo sviluppo non solo di siti

web,ma anche delle moderne applicazioni Web.
WebML è un linguaggio concettuale in grado di supportare tutte le attività e le prospettive della modellazione delle applicazioni web Data-Intensive.
Il Web Modelling Language mette a disposizione specifiche grafiche,ma anche formali,inserite in un processo di modellazione completo che può essere assistito da CASE tools specifici.
I principali obiettivi del processo di modellazione WebML sono:

1. Esprimere la struttura di un'applicazione web mediante una descrizione di alto livello.

2. Mettere a disposizione più punti di vista dello stesso contenuto.

3. Separare il contenuto informativo dalla sua composizione in pagine,dalla navigazione e dalla presentazione che possono essere definite e sviluppate indipendentemente.

4. Memorizzare le meta-informazioni raccolte durante il processo di modellazione in un Data-Repository che può essere utilizzato durante la fase di operabilità dell'applicazione per generare dinamicamente pagine Web.

5. Modellare esplicitamente gli utenti ed i gruppi di utenti allo scopo di permettere la specifica di politiche di personalizzazione.

6. Permettere la specifica di operazioni per la manipolazione dei dati allo scopo di modificare il contenuto dell'applicazione o di interagire con servizi esterni.

WebML può essere usato sia con un approccio "top-down" che con uno "bottom-up".Nel caso top-down le sorgenti di dati (ad esempio un database online) sono definite insieme con l'applicazione;nel

caso bottom-up,le sorgenti di dati esistono a priori e devono essere integrate con il processo.Nello scenario bottom-up si possono applicare gli stessi principi ma il processo di modellazione deve includere un mapping esplicito delle strutture dati concettuali in WebML nelle sorgenti di dati fisiche preesistenti,la cui struttura e le cui primitive d'accesso possono influenzare o porre dei vincoli al processo di modellazione WebML.

I modelli WebML

WebML assume un approccio step-by-step nella specifica delle applicazioni Web Data-Intensive,dove ogni step riguarda una prospettiva distinta della modellazione.Le tre prospettive ortogonali sono:

Modello strutturale : esprime l'organizzazione concettuale dei dati in termini di *Entità* e di *Relazioni* rilevanti.Tale modello è un sottoinsieme del modello Entity-Relationship e dei diagrammi UML delle classi.Gli elementi fondamentali del modello strutturale sono le *entità*,definite come contenitori di dati, e le *relazioni* ,definite come collegamenti semantici tra le entità.Come nel modello E/R le entità hanno proprietà con nomi,chiamate *attributi*, con un proprio tipo associato;le entità possono essere organizzate in *gerarchie di generalizzazione*e le relazioni possono essere ristrette da *vincoli di cardinalità*.

Tuttavia l'aspetto grafico è molto diverso da quello dell'E/R infatti,come si può vedere dalla figura 3.2,la grafica è quella dei diagrammi delle classi UML anche se nelle classi non sono presenti i metodi in quanto le possibili operazioni sui dati vengono modellate in seguito.

Allo scopo di soddisfare il requisito di esprimere informazione derivata o calcolata,il modello strutturale offre la pos-

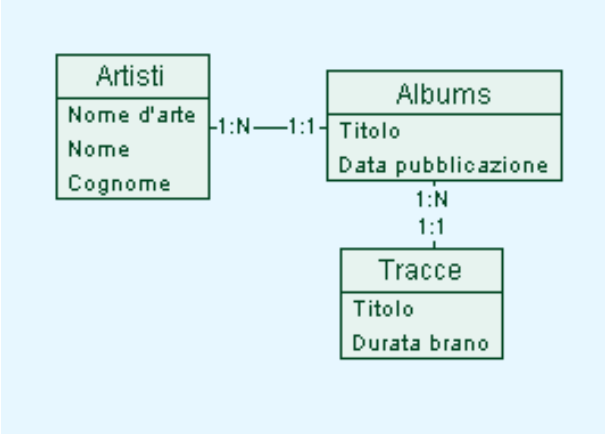

Figure 3.2: Esempio di modello strutturale in WebML

sibilità di supportare la *derivazione*.
La derivazione è il processo di aggiungere dati rindondanti allo schema strutturale allo scopo di aumentarne l'espressività e definire viste diverse e raggruppamenti dello stesso dato.La derivazione viene espressa attraverso delle queries ,in un linguaggio simile ad OQL,applicate agli elementi dello schema strutturale.Ogni elemento del modello strutturale (entità,attributi,relazioni) può essere derivato attraverso le queries.
Le operazioni più frequenti di derivazione sono:l'importazione di attributi da un'entità all'altra ,la definizione di attributi calcolati o la creazione di relazioni derivate concatenando o vincolando le relazioni già esistenti.

Modello dell'Ipertesto: descrive uno o più ipertesti che possono essere pubblicati dall'applicazione.Ogni ipertesto ,cioè ogni insieme di pagine che realizzano un proposito ben definito dell'applicazione,viene immagazzinato in un costrutto di modularizzazione,detto *site-view*.
Una vista di sito è un ipertesto che assolve ai requisiti di una particolare classe di utenti,offrendo interfacce navigazionali per accedere all'informazione,manipolarla ed at-

tivare servizi.
Sulla base dello stesso modello strutturale possono essere definite più viste di sito per soddisfare le necessità di diversi gruppi di utenti o per riarrangiare la composizione delle pagine allo scopo di adattare l'applicazione alle peculiarità dei vari dispositivi di accesso.
La descrizione di ogni vista di sito consiste di due sottomodelli:

Modello di composizione: concerne la definizione delle *pagine* e della loro organizzazione interna in termini di *content units* elementari.Le *pagine* sono i contenitori dell'informazione fornite in un certo istante all'utente.Le *units* sono elementi dal contenuto atomico usati per rendere disponibile l'informazione descritta nel modello strutturale.
In WebML sono predefiniti sei tipi di *content units* per la composizione delle pagine:

- *Data units*:sono usate per visualizzare l'informazione di un singolo oggetto,cioè l'istanza di un'entità (ad esempio,un album musicale-figura 3.3-).

Figure 3.3: Esempio di Data Unit

- *Multi-Data units*:vengono utilizzate per visualizzare l'informazione di un insieme di oggetti,cioè tutte le istanze di un'entità(ad esempio,un insieme di album musicali-figura 3.4-).

Figure 3.4: Esempio di Multi-Data Unit

- *Index units*:sono usate per mostrare una lista di oggetti senza presentare informazioni dettagliate per ogni oggetto della lista (ad esempio,una lista di album-figura 3.5-).

Figure 3.5: Esempio di Index Unit

Esistono due varianti di Index Units:

- *Multi-Choice Index units*:variante delle *Index units* in cui ad ogni elemento della lista viene associato un checkbox permettendo all'utente di compiere più di una scelta (ad esempio,la scelta di più albums-figura 3.6-).
- *Hierarchical Index units*:variante delle *Index units* in cui gli oggetti della lista sono organizzati in un albero multi-livello (figura 3.7).

Figure 3.6: Esempio di Multi-Choice Index Unit

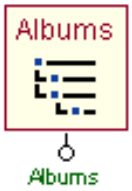

Figure 3.7: Esempio di Hierarchical Index Unit

- *Entry units*:utilizzate ogni volta che si vogliono fare immettere dei dati all'utente utilizzando una Web form (ad esempio,far inserire all'utente l'artista sul quale vuole avere informazioni,figura 3.8).

Figure 3.8: Esempio di Entry Unit

- *Scroller units*:mostrano comandi per accedere agli elementi di un insieme ordinato di oggetti (ad esempio,per scorrere gli albums,figura 3.9).

Figure 3.9: Esempio di Scroller Unit

Il contenuto delle *content units* viene "estratto" dalle entità specificate nello schema strutturale;dunque ogni *content unit* è associata con un'entità sottostante,fatta eccezione per le *Entry units* le quali non sono definite su nessuna entità poichè esse non pubblicano il contenuto del database ma accettano l'input dell'utente.
La specifica dell'entità sottostante indica il tipo di oggetto dal quale deriva il contenuto dell'unità,ma non identifica le istanze specifiche di quell'entità che devono essere utilizzate.Per questo motivo,quando è appropriato,le unità possono essere associate con un *selettore*,cioè un insieme di condizioni che determinano le attuali istanze da usare come contenuto dell'unità.Il selettore può essere definito su un attributo dell'entità o su una relazione.Ad esempio,una *index unit* definita sull'entità Album può avere un selettore costruito sulla relazione tra autore ed album allo scopo di restringere l'insieme degli album da mostrare ai soli album di quell'autore specifico (figura 3.10).

Figure 3.10: Esempio di selettore definito su una relazione

Modello di navigazione: descrive i links tra le *pagine* e le *content units* per formare l'ipertesto.Tali links assicurano che la necessaria informazione relativa al contesto sia disponibile per determinare il contenuto delle pagine e fornire meccanismi per l'individuazione dell'informazione.
Un link può avere i seguenti propositi:

1. Spostare l'attenzione dell'utente da una pagina all'altra.

2. Passare informazioni da un'unità all'altra.
3. Essere associati ad operazioni per produrre effetti specifici (ad esempio,l'esecuzione di un'operazione di modifica).

L'informazione trasportata dai links è chiamata *contesto navigazionale* o più semplicemente *contesto*.
I links che trasportano l'informazione del contesto sono chiamati *links contestuali*:essi collegano due unità dove il conteuto dell'unità di destinazione del link è determinato sulla base dell'informazione proveniente dall'unità sorgente.I links contestuali possono essere *intra-pagina*,se collegano unità della stessa pagina,oppure *inter-pagina*,se collegano unità di pagine diverse.
I links che non trasportano l'informazione del contesto sono chiamati *links non-contestuali*:essi collegano due pagine semanticamente indipendenti.
L'informazione del contesto serve per poter definire il selettore dell'unità di destinazione del links:ad esempio (figura 3.11) ,nel link tra un autore ed i propri albums verràtrasportata l'informazione dell'identificativo (OID) dell'autore allo scopo di poter definire il selettore sulla relazione autore-album.

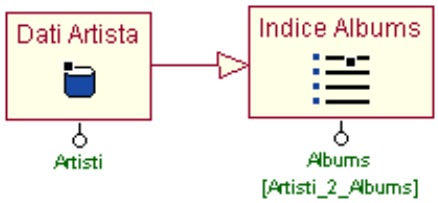

Figure 3.11: Esempio di link contestuale

I links WebML,sia contestuali che non,possono essere di tre tipi:

1. *Links normali*:vengono navigati dall'utente cliccandovi.
2. *Links di trasporto*:trasportano informazioni,ma non interagiscono con l'utente,cioè non compaiono nella pagina e non possono essere cliccati.
   Graficamente i links di trasporto si distinguono perchè rappresentati con una linea tratteggiata,anzichè continua.
3. *Links automatici*:trasportano il contesto di default(l'identificatore della sorgente)ma collegano pagine che vengono caricate automaticamente senza l'interazione con l'utente.

Modello di presentazione :esprime il layout e l'aspetto grafico delle pagine
indipendentemente dal dispositivo d'accesso attraverso una sintassi XML astratta.
Le specifiche di presentazione possono essere specifiche della pagina o generiche.Nel primo caso tali specifiche riguardano una determinata pagina ed includono riferimenti espliciti al contenuto della pagina (ad esempio,la pagina di un artista nella home page di un sito);nel secondo caso esse sono basate su modelli predefiniti indipendenti dal contenuto specifico della pagina ed includono riferimenti ad elementi generici del contenuto(ad esempio,indicano il layout di tutti gli attributi di un generico oggetto della pagina).

I modelli strutturale,di composizione,di navigazione permettono la descrizione di applicazioni web "read-only".
Essi possono essere estesi con specifiche di **content management** e con l'integrazione di servizi esterni,attraverso l'introduzione di *operazioni.*Tali operazioni vengono modellate all'interno delle *site views* mediante unità particolari dette *operation units.*
Un'*operation unit* è un'unità WebML che realizza un'azione sui

dati,come creare,modificare o cancellare un oggetto.
Un'operation unit può ricevere links da content units o da altre operation units può avere tre tipi di links in uscita(un link di trasporto,un OK link ,nel caso di successo dell'operazione,ed un KO link nel caso di insuccesso)che possono puntare a pagine,a content units all'interno delle pagine o ad altre operazioni.
WebML fornisce alcune operation units predefinite:

- *Create unit*:permette di creare una nuova istanza di un'entità sulla quale è definita.Generalmente un'operazione di creazione viene attivata da un link proveniente da un'entry unit,utilizzata per permettere all'utente di inserire i valori degli attributi del nuovo oggetto.L'icona della create unit è mostrata in figura 3.12.

Figure 3.12: Esempio di create unit

- *Delete unit*:permette di distruggere un'istanza dell'entità sulla quale è definita.La relativa icona è mostrata in figura 3.13.

Figure 3.13: Esempio di delete unit

- *Modify unit*:permette di modificare il contenuto dell'applicazione Web.
  Una modify unit presenta sempre due sorgenti di input:un'entry

unit,per mezzo della quele l'utente specifica i valori da utilizzare nella modifica degli attributi dell'oggetto,ed una data unit che fornisce l'oggetto da modificare.
La relativa icona èmostrata in figura 3.14.

Figure 3.14: Esempio di modify unit

- *Connect unit*:permette di collegare istanze di due entità ,mediante la creazione di una relazione tra esse.Tale unit è definita sulla base della relazione tra le entità interessate.
  La relativa icona è mostrata in figura 3.15.

Figure 3.15: Esempio di connect unit

- *Disconnect unit*:permette di cancellare la relazione tra due oggetti o tra un insieme di oggetti.La relativa icona è mostrata in figura 3.16.

Figure 3.16: Esempio di disconnect unit

WebML è fornito di uno strumento di CASE,chiamato **WebRatio**.WebRatio Site Development Studio fornisce un'interfaccia grafica per costruire in modo visuale i diagrammi E/R-UML che descrivono la

struttura dell'informazione gestita dall'applicazione ed i diagrammi degli ipertesti delle viste di sito operanti su tale informazione.
Il cuore di WebRatio è un generatore di codice basato su XML ed XSL in grado di trasformare le specifiche WebML,codificate in XML,nel codice finale dell'applicazione per piattaforme J2EE e Microsoft.NET.
Il codice generato copre tutti gli aspetti di un'applicazione Web dinamica:le queries per l'estrazione delle informazioni dalle fonti dati,il codice per la logica di business dell'applicazione ed i template di pagina per la costruzione dinamica delle interfacce utente.
Un aspetto innovativo di WebRatio è la cura particolare dedicata gli aspetti di presentazione,cioè alla disposizione dei contenuti nella pagina e alla veste grafica.

Analisi di WebML sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali di WebML descritti precedentemente permettono di verificare se il Web Modelling Language soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,WebML introduce una chiara separazione tra i livelli di struttura,ipertesto e presentazione,come descritto precedentemente,permettendo un mapping tra i livelli esplicito e flessibile.

- Dal punto di vista dei *aspetti*,l'aspetto strutturale viene considerato e trattato a tutti i livelli.
  L'aspetto comportamentale viene trattato a livello di presentazione ma anche a livello di ipertesto mediante l'introduzione di operazioni all'interno delle site-views.

- Dal punto di vista delle *fasi*,l'approccio di WebML è concentrato sulla modellazione concettuale dell'ipertesto,basandosi sul modello E/R per quella dei dati.Tuttavia,nonostante l'inclinazione del linguaggio verso la modellazione concettuale,collegati a WebML vengono affrontati anche problemi riguardanti l'implementazione di Data-intensive Web Applications(ad esempio lo sviluppo di CASE tool che supportino WebML).

3.2 Linguaggi di modellazione web nell'ambito dei *Database systems*

3.2.1 RMM

La metodologia RMM(Relationship Management Methodology) [11] è influenzata dal modello E/R e da HDM.Il nome "relationship management" sottolinea come l'ipermediale venga considerato come mezzo per la gestione delle relazioni tra gli oggetti contenenti l'informazione.
Come gli stessi autori indicano in [11],la metodologia RMM risulta maggiormente indicata per le classi di applicazioni che presentano una struttura regolare del dominio di interesse,cioè una struttura con classi di oggetti,relazioni definibili tra tali classi e molteplicità di istanze degli oggetti di ogni classe.
RMM è dunque utile per le applicazioni con una struttura regolare ma anche con un'elevata volatilità dei dati,caratteristiche che la maggior parte delle applicazioni web presentano.

Tale metodologia si caratterizza per la definizione di una sequenza raccomandata di passi da seguire,per la presenza di formalismi per le strutture d'accesso,per la maggior importanza data alla rappresentazione grafica e per una dettagliata procedura di design e sviluppo dell'ipermediale.
La metodologia *RMM(Relationship Management Methodology)*

prevede sette passi per il design e la costruzione di progetti ipermediali;alla base della metodologia vi è il modello *RMDM(Relationship Management Data Model)* il quale rappresenta il risultato dei primi tre passi della metodologia.

Relationship Management Data Model (RMDM)

Un modello dei dati è,in generale,un insieme di oggetti logici usati per creare l'astrazione di una parte del "mondo reale".
I modelli utilizzati per i databases risultano inadeguati per cogliere le peculiarità del mondo ipermediale,come la navigazione,dunque la metodologia RMM propone un nuovo modello:il modello *RMDM(Relationsh Management Data Model).*
RMDM mette a disposizione un linguaggio per la descrizione degli oggetti contenti l'informazione e dei meccanismi di navigazione nell'ambito delle applicazioni ipermediali.
La figura 3.17 mostra le primitive di modellazione RMDM.

Nella parte alta della figura vi sono le *primitive di dominio*le quali modellano l'informazione riguardante il dominio dell'applicazione.Le *entità* ed i loro *attributi* rappresentano oggetti astratti,come un account bancario o fisici,come una persona.
Le *relazioni associative*,che possono essere uno-uno o uno-molti,descrivono associazione tra i diversi tipi di entità.Come nella modellazione dei database,le relazioni molyi.molti vengono suddivise in due relazioni uno-molti.
Piochè le entità possono avere molti attributi di natura diversa (ad esempio,informazioni sul salario,dati biografici,fotografie),risulta poco pratico ed inefficente rappresentare tutti gli attributi di un'istanza di un'entità.
Per questo motivo gli attributi vengono raggruppati in *slices*,la cui rnotazione grafica è rappresentata nel centro della figura 3.17.
Ad esempio l'entità persona con gli attributi nome,anni,foto e

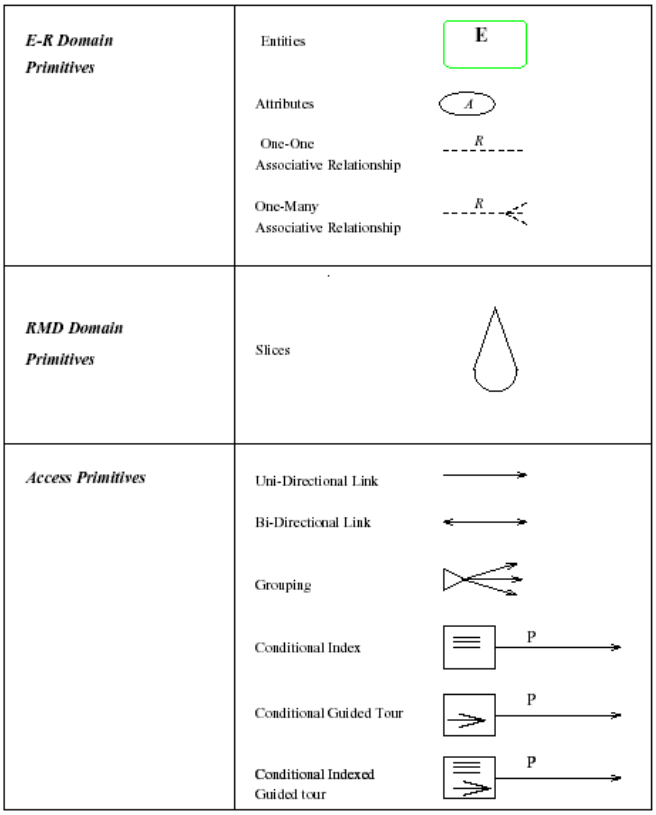

Figure 3.17: Le primitive del modello *RMDM(Relationship Management Data Model)*

biografia ha una slice generale ,contenente nome,anni e foto, ed una slice biografica contenente nome e biografia.Dunque ogni istanza dell'entità persona sarà rappresentata da due slices e ,se l'applicazione lo supporta,l'utente potrà scegliere quale delle due vedere.
La navigazione è supportata da RMDM mediante le sei *primitive di accesso* rappresentate in basso nella figura 3.17.

- I *links unidirezionali e bidirezionali* sono usati per specificare l'accesso tra le slices di un'entità.Èimportante sottolineare che tali links possono essere utilizzati solo per navigari

all'interno dei confini di un'entità ,cioè tra le slices che la compongono.RMDM supporta la navigazione attraverso le entità attraverso i costrutti come *indici*,*cammini guidati* e *raggruppamenti*.

- Un *indice* ha la funzione di una tabella dei contenuti per una lista di istanze e permette l'accesso diretto ad ogni elemento della lista.

- Un *cammini guidato*implementa un percorso lineare tra un insieme di elementi ,permettendo all'utente di muoversi sul percorso nei due possibili sensi.

- Il costrutto del *raggruppamento* è un meccanismo simile ad un menu che permette l'accesso ad altre parti dell'applicazione.Un tipico esempio di raggruppamento è l' "opening screen" di molte applicazioni che ha lo scopo di permettere l'accesso ad altri indici,cammini guidati etc..Gli indici un tipo speciale di raggruppamenti.

Ai links sono spesso associate delle condizioni o *predicati logici*che qualificano indici e cammini guidati determinando quali istanze di un'entità sono accessibili dal costrutto.
Ad esempio,la figura 3.18-a mostra un cammino guidato condizionato di tutti i professori associati.Il predicato logico (Corpo docenti(ordine="associato")) indica che solo le istanze dell'entità corpo docenti il cui attributo ordine è "associato" partecipano al cammino guidato.La parte destra della figura mostra un'istanza di tale cammino guidato.
Analoghe considerazioni valgono per l'indice condizionato di figura 3.18-b e per la figura 3.18-c che combina i due costrutti allo scopo di ottenere una struttura d'accesso più efficente.
Èimportante notare che un diagramma RMDM descrive come gli utenti possono navigare all'interno dell'applicazione,in contrasto con un diagramma Entity-Relationship che rappresenta il

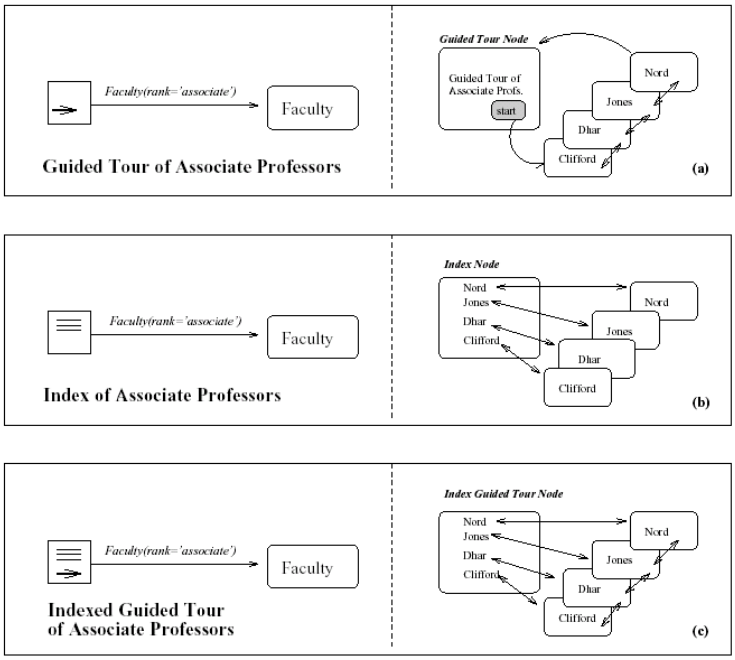

Figure 3.18: Esempi di costrutti di accesso RMDM

design di un database.

Relationship Management Design Methodology (RMM)

La metodologia RMM è mostrata graficamente nella figura 3.19 nel contesto di un ciclo di vita completo del software.RMM è centrato sulle fasi di design,sviluppo e costruzione ed individua sette passi successivi per la realizzazione di tali fasi.

Passo 1 :Design dell'E/R.
Il primo passo di design è rappresentare il dominio informativo di un'applicazione mediante il modello Entity-Relationship(E/R). Tale fase del processo di design rappresenta uno studio delle entità e delle relazioni del dominio dell'applicazione considerate rilevanti.Tali entità ed associazioni formano la base dell'applicazione e molti di essi saranno presenti nell'applicazione finale come nodi

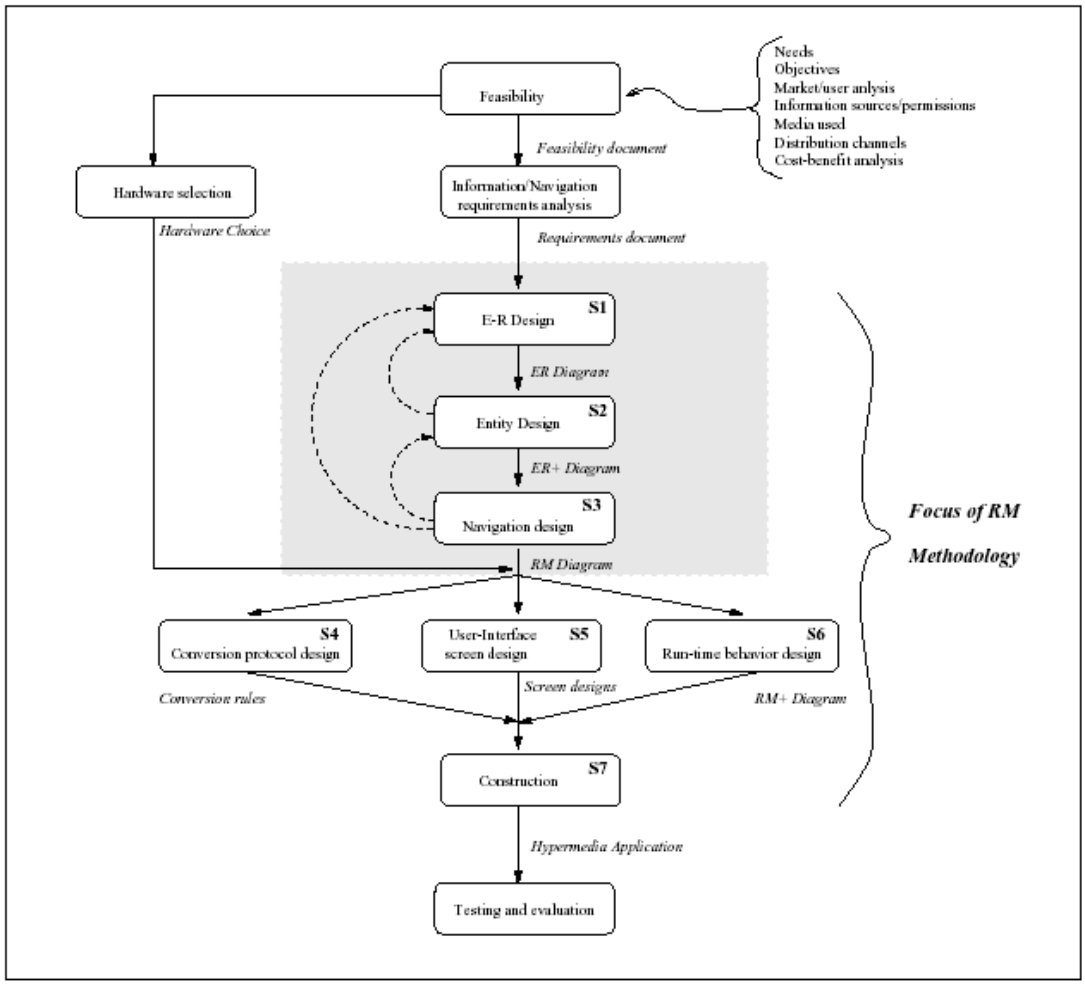

Figure 3.19: La metodologia RMM

e links.
L'analisi del dominio usando un approccio E/R aiuta ad identificare relazioni importanti attraverso le quali può essere supportata la navigazione.

Passo 2 :Design delle slices.
Questa fase determina come l'informazione delle entità scelte verrà presentata agli utenti e come essi potranno accedervi.Ogni

entità viene suddivisa in slices con un preciso significato e queste sono organizzate in una rete.Ad esempio la figura 3.20 mostra l'entità Corpo docenti suddivisa in quattro slices contenenti informazioni generali,una corta biografia,interessi di ricerca e un video-clip.
I links che rappresentano le connessioni tra le slices vengono chiamati *links strutturali*per differenziarli dalle relazioni associative dell'E/R.Infatti i links strutturali connettono parti di informazione all'interno della stessa istanza di entità,mentre le relazioni associative connettono diverse istanze di entità.Dal punto di vista navigazionale tale distinzione è molto significativa in quanto quando l'utente percorre un link associativo il contesto dell'informazione cambia ,mentre quando percorre un link strutturale il contesto rimane sempre limitato alla stessa istanza.
La fase di design delle slices prevede dunque:

- La suddivisione dell'entità in slices:le slices devono unire elementi contenenti informazione che sono in relazione tra loro , ma non dovrebbero contenere troppa informazione.

- La scelta di una slice come *head* dell'entità:tale scelta richiede un'analisi del dominio,ad esempio in figura 3.20 la slice generale è stata scelta come *head* poich è è la pi ù rappresentativa.

- L'interconnessione delle slices:vi è la necessitàdi avere links che collegano la slice *head* con le altre e viceversa.

- Associare etichette ai links:la scelta delle etichette da associare ai links è molto importante; nella figura 3.20 sono state scelte etichette che manifastano esplicitamente la natura delle slices collegate dal link.

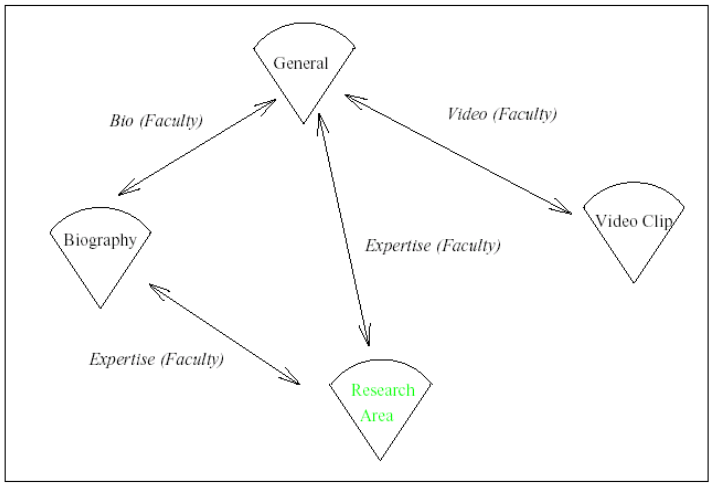

Figure 3.20: Diagramma delle slices per un corpo docente

Passo 3 :Design della navigazione.
In questa fase viene definito il cammino che permetterà la navigazione.Ogni relazione associativa che compare nel modello E/R viene analizzata e ,se compatibile con la specifica dei requisiti,può diventare accessibile per la navigazione:ciò significa che sarà sostituita da una o piùstrutture d'accesso RMDM (indici,cammini guidati o raggruppamenti).
Il nome della relazione viene usato come condizione nelle corrispondenti strutture d'accesso per indicare quali istanze delle entità devono essere connesse.
Di default,le strutture d'accesso accedono alla slice *head* dell'entità.
Al termine della fase di design della navigazione il diagramma E/R viene trasformato in un diagramma RMDM che descrive tutte le strutture d'accesso del sistema.

Passo 4 :Conversion Protocol Design.
Viene utilizzato un insieme di regole di conversione per trasformare ogni elemento del diagramma RMDM in un oggetto della

piattaforma di destinazione.Tale fase viene solitamente realizzatamanualmente dai programmatori.

Passo 5 :Design dell'interfaccia utente.
Riguarda il design del layout di ogni oggetto che appare nel diagramma RMDM ottenuto al passo 3 ,cioè il design di bottoni ,nodi ed indici.

Passo 6 :Design del comportamento run-time.
Durante questa fase vengono prese decisioni riguardo a come devono essere implementati meccanismi di navigazione o di mantenimento dello storico;inoltre gli sviluppatori considerano la volatilità e la dimensione del dominio per decidere se i contenuti dei nodi e gli endpoints dei links devono essere costruiti durante lo sviluppo dell'applicazione o calcolati dinamicamente a runtime.

Passo 7 :Costruzione e testing.
Tale fase corrisponde a quella dei tradizionali progetti software;nelle applicazioni ipermediali è necessaria particolare attenzione nel testare tutti i cammini navigazionali.

Analisi di RMM sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali della metodologia RMM descritti precedentemente permettono di verificare se la Relationship Management Methodology soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,RMM riconosce tutti i tre livelli di contenuto,ipertesto e presentazione.Il livello di con-

tenuto viene modellato separatamente mentre il livello di presentazione viene definito insieme all'ipertesto. Viene introdotto un particolare formalismo,il modello *Relationship Management Data Model(RMDM)* decritto precedentemente,utilizzando il modello E/R per il livello di contenuto e concetti proprietari influenzati da HDM per il livello di ipertesto e di presentazione.Il concetto di *slice*,descritto precedentemente,viene utilizzato per effettuare il mapping tra il livello di contenuto e quello di ipertesto nel quale vengono raggruppati insieme gli attributi delle entità dell'E/R o le slices precedentemente definite.Vengono messi a disposizione patterns per la navigazione come indici e cammini guidati.

- Dal punto di vista dei *aspetti*,vengono considerati solo aspetti strutturali a tutti i livelli.

- Dal punto di vista delle *fasi*,RMM specifica un processo di sviluppo con le fasi iniziali di specifica dei requisiti e modellazione del contenuto tramite l'E/R,seguiti da altri passi interattivi che definiscono una metodologia completa fino alla costruzione e testing dell'applicazione.

3.2.2 Araneus

La metodologia per il design Web Araneus [12] rappresenta un processo di design sistematico per organizzare e gestire grandi quantità di dati in un ipertesto web.Araneus si basa principalmente sul caso in cui i dati da pubblicare nel sito sono memorizzati in un database(relazionale o object-oriented).Infatti i DBMS forniscono una tecnologia robusta per la gestione di grandi collezioni di dati in modo flessibile ed efficente;Araneus si propone di utilizzare tale metodologia per la gestione ed il mantenimento del web.

La metodologia Araneus (figura 3.21) è basata su due processi distinti ma interconnessi tra loro:

- *Il processo di modellazione del database*
- *Il processo di modellazione dell'ipertesto*

Per ogniuno di questi processi si distinguono una *fase di modellazione concettuale* ed una *fase di modellazione logica*,analogamente a come avviene nell'ambito dei databases.Ciò permette di isolare elementi del sistema da prospettive di livello più alto e facilita un'eventuale processo di ricostruzione.
Infine,una volta generati lo schema logico del database e quello dell'ipertesto la struttura dell'ipertesto deve essere mappata sulla struttura del database ,allo scopo di creare fisicamente le pagine HTML:per fare ciò gli attributi delle pagine vengono associati con i corrispondenti attributi provenienti dalle tabelle o dalle viste del database.
Dalla figura 3.21 si può inoltre notare che viene dedicata una fase specifica (fase 5) alla modellazione della presentazione,cioè il layout in base al quale verrà visualizzata l'informazione nelle pagine.

Allo scopo di supportare tale metodologia è stato sviluppato un CASE tool,al quale è stato dato il nome di *HOMER*,il quale supporta tutte le fasi dello sviluppo permettendo di passare da una fase alla successiva interagendo con un'interfaccia grafica ma sopratutto tale CASE tool genera automaticamente tutto il codice necessario per lo sviluppo di un sito web.

Fasi 1 e 2 : Il processo di modellazione del database

L'approccio di Araneus eredita da RMM il fatto che lo schema E/R viene considerato come base per la modellazione dell'ipertesto.

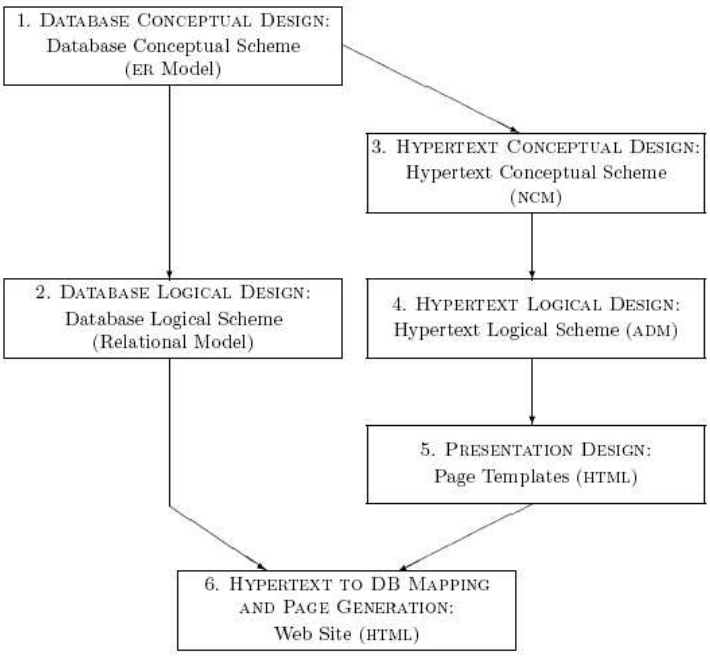

Figure 3.21: La metodologia Aranues

Il database contente i dati da pubblicare può essere sia preesistente che da implementare.Se il database deve essere esplicitamente creato,risultano necessarie entrambe le fasi di modellazione concettuale e logica.
Nel caso di database preesistente tali fasi possono essere omesse considerando valido lo schema concettuale del dominio applicativo,anche se tale schema può non essere disponibile e quindi viene resa neccessaria una fase di reverse-engineering per ottenere uno schema concettuale a partire dal database esistente.

Il punto di partenza del processo di modellazione del database è la costruzione di uno *schema concettuale E/R*,considerato come standard nella modellazione dei databases.Lo schema E/R rappresenta una descrizione concettuale del dominio applicativo in-

dipendente da qualunque scelta implementativa,basata su *entità* e su *relazioni* tra le entità.
Un esempio a cui far riferimento durante la descrizione della metodologia completa può essere la modellazione di un sito web del Dipartimento dell'Università,il cui schema E/R è rappresentato in figura 3.22.Si suppone che lo schema E/R sia stato disegnato utilizzando le tecniche standard e che sia disponibile lo schema relazionale di figura 3.23.

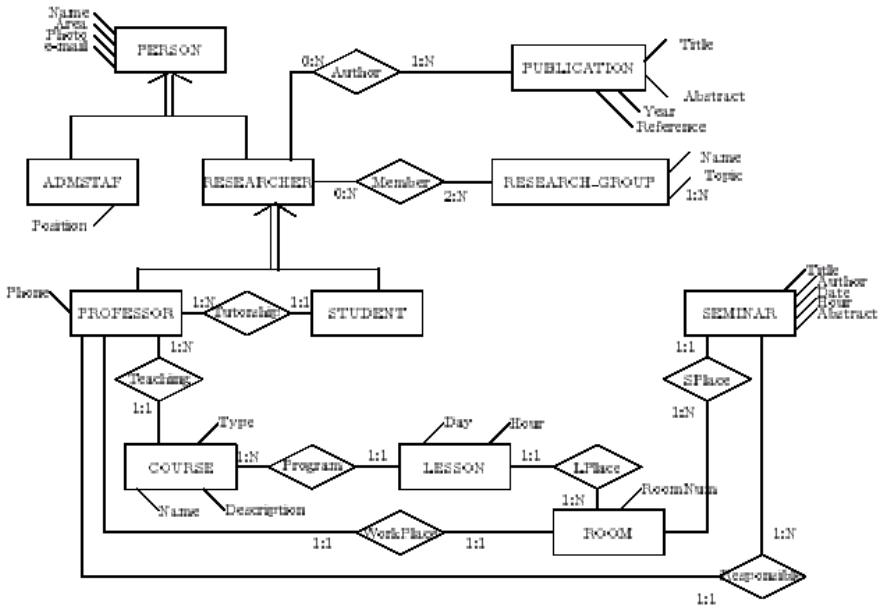

Figure 3.22: Esempio di schema E/R:lo schema di un dipartimento

Fase 3 : Modellazione concettuale dell'ipertesto : il modello NCM

La modellazione concettuale dell'ipertesto ha lo scopo di descrivere come i concetti del dominio applicativo devono essere organizzati in forma ipertestuale;tale attività è indipendente dall'implementazione fisica e si concentra su due aspetti fondamentali :

1. Decidere quali entità concettuali faranno parte dell'ipertesto cioè quali corrisponderanno ai "nodi" dell'ipertesto.

2. Scegliere i cammini tra le entità da navigare,cioè come è possibile esplorare la rete dei nodi.

A questo livello vengono descritte anche le strutture di accesso all'ipertesto,cioè come le entità devono essere aggregate allo scopo di fornire un'organizzazione gerarchica dei concetti che devono essere esplorati dall'utente.

La descrizione delle proprietà concettuali di un ipertesto implica tre principali attività:

- Descrivere le entità cocncettuali che fanno parte dell'ipertesto cioè i *nodi dell'ipertesto*.Ogni entitàdescrive un oggetto del mondo reale presente nell'ipertesto e può essere vista come l'aggregazione di più attributi;ciò è coerente con il fatto che,come un database,un ipertesto è basato su un modello del dominio sottostante.Non tutte le entità devono anche diventare nodi dell'ipertesto;ad esempio nella figura 3.22le entità PROFESSOR e PUBBLICATION saranno molto probabilmente nodi dell'ipertesto,mentre l'entità ROOM probabilmente non diventerà un nodo.

```
PROFESSOR(Name, Area, Photo, e-mail, Phone, RoomNum)
STUDENT(Name, Area, Photo, e-mail, Tutor)
ADMSTAFF(Name, Area, Photo, e-mail, Position)
COURSE(Name, Type, Description, Instructor)
LESSON(CourseName, Day, Hour, RoomNum)
SEMINAR(Title, Author, Date, Hour, Abstract, Responsible)
PUBLICATION(Title, Abstract, Year, Reference)
RESEARCH-GROUP(Name, Topic)
PERSON-IN-GROUP(Name, Group)
PUBLICATION-AUTHORS(Name, Title)
```

Figure 3.23: Esempio di schema relazionale:lo schema relazionale del dipartimento

- Descrivere i cammini ipertestuali per navigare tra le enità.Rispettando le tradizionali relazioni E/R,la descrizione di cammini attraverso le entità richiede l'introduzione di una notazione di *direzione* ,poich è in un ipertesto una navigazione va sempre da un nodo sorgente ad uno destinazione.

- Descrivere quali *cammini d'accesso*deve mettere a disposizione l'ipertesto a partire da un nodo sorgente allo scopo di localizzare l'informazione di interesse;ciò corrisponde ad organizzare le entità in una gerarchia di concetti da esplorare da parte dell'utente.

Il modello dei dati utilizzato per descrivere lo schema concettuale dell'ipertesto viene chiamato *Navigation Conceptual Model (NCM).*
Il modello NCM si ispira ad RMM,anche se elimina tutti gli elementi non concettuali.Vi sono due classi principali di costrutti in NCM:i *nodi* ed i *links*,rappresntati graficamente in figura 3.24.
Vi sono tre diversi tipi di nodi:

- *Macroentità*:una macroentità è un nodo corrispondente ad una classe di oggetti nel dominio dell'applicazione;le macroentità hanno attributi,che possono essere semplici o complessi;ogni attributo ha associata una cardinalità .Per ogni macroentità deve essere indicata una *chiave descrittiva*,cioè un insieme di attributi con due proprietà:è una chiave per le istanze della macroentità ed è esplicativo riguardo alla corrispondente istanza,cioè l'utente può cogliere direttamente il significato dei valori degli attributi che lo compongono.
  Ad esempio nell'esempio del dipartimento di figura 3.22 una chiave descrittiva per l'entità SEMINARIO è formata dagli attributi Titolo ed Autore del seminario.

- *Nodi di Unione*:Un nodo di unione corrisponde all'unione di più nodi dell'ipertesto (possibilmente eterogenei). I nodi

di unione sono utilizzati in NCM per modellare tipi corrispondenti all'unione di diverse macroentità.

- *Nodi di Aggregazione*:I nodi di aggregazione,insieme ai links di aggregazione,sono le primitive che permettono di modellare le strutture d'accesso all'ipertesto.Un nodo di aggregazione descrive un'aggregazione o una classificazione di macroentità.Le aggregazioni possono corrispondere alle gerarchie(ad esempio, PERSONA come aggregazione di PROFESSORE e STUDENTE) o possono essere introdotte esplicitamente per meglio organizzare l'informazione all'interno dell'ipertesto.

Vi sono due tipi diversi di links:

- *Relazioni direzionate*:Una relazione direzionata descrive come sia possibile navigare in un ipertesto da un nodo sorgente ad uno destinazione basandosi sulle loro relazioni concettuali.I vincoli di cardinalità associati con il nodo sorgente vengono usati per modelllare il numero delle istanze del nodo destinazione associato con un'istanza della sorgente.Un tipo speciale di relazioni direzionate,chiamate simmetriche,vengono utilizzate per indicare che la navigazione tra i due nodi può avvenire nei due versi.

- *Links di Aggregazione*:Un link di aggregazione viene utilizzato per connettere un nodo di aggregazione con i corrispondenti nodi dell'ipertesto.

L'approccio considerato è dunque di tipo "bottom-up",nel senso che i costrutti dell'ipertesto vengono derivati dai corrispondenti costrutti dell'E/R.

Nel passaggio dallo schema E/R allo schema concettuale dell'ipertesto si possono individuare alcune fasi metodologiche fondamentali:

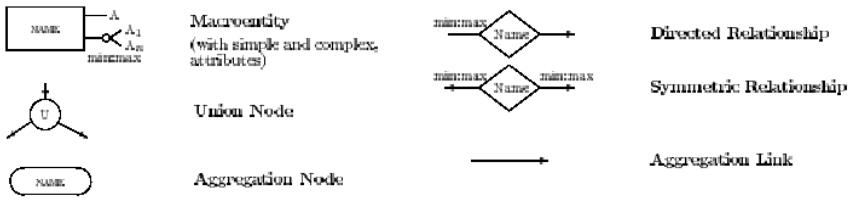

Figure 3.24: Rappresentazione grafica dei costrutti NCM

- *Modellazione delle macroentità*:le macroentitàsono descrizioni intensionali delle classi di oggetti del mondo reale,esse indicano la più piccola parte di informazione autonoma che può avere un'esistenza indipendente nell'ipertesto.Esempi di macroentità nell'esempio del dipartimento sono lo STUDENTE ed il CORSO.Le macroentità derivano in modo naturale dalle entità E/R o,in alcuni casi,dalle viste dell'E/R.

- *Modellazione dei nodi di unione*:i nodi di unione corrispondono all'unione di macroentità diverse.Essi vengono ottenuti principalmente mappando le gerarchie E/R o possono essere introdotti esplicitamente per modellare la navigazione che coinvolge macroentittà diverse.ad esempio,considerando la parte della figura che parla delle pubblicazioni e degli autori,poichè un autore putò essere sia un professore che uno studente,per modellare tale relazione è necessario un tipo corrispondente all' unione di studente e professore.

- *Modellazione delle relazioni direzionate*:le relazioni direzionate modellano le navigazioni tra i nodi dell'ipertesto e si basano sulle associazioni concettuali.In generale le relazioni direzionate derivano dalle relazioni E/R.Ad esempio,nell'ambito del dipartimento la relazione direzionata tra SEMINARIO e PROFESSORE nello schema NCM modella il fatto che è possibile navigare dal seminario ai suoi organizzatori (e

non viceversa);utilizzando una relazione direzionata simmetrica la navigazione sarebbe possibile in entrambe le direzioni,quindi anche dal professore ai seminari organizzati.

- *Modellazione delle aggregazioni*:i nodi di aggregazione ed i links di aggregazione vengono utilizzati per organizzare e classificare l'ipertesto allo scopo di imporre una struttura gerarchica.A volte un'aggregazione pu ò essere solo parziale,cio è solo una parte delle istanze della macroentit àdevono essere aggregate:ciò pu ò essere modellato in NCM associando delle *labels* ai links di aggregazione;ogni label ha associato un predicato ed è usata per specificare che solo le istanze che soddisfano tale predicato sono considerate parte dell'aggregazione.Nell'esempio del dipartimento (figura 3.25)si può pensare di aggregare con il nome di EDUCAZIONE le macroentità CORSO e PROFESSORE, tuttavia è ragionevole distinguere i corsi graduati da quelli non-graduati:ciò è possibile modellando due links con associate due labels diverse.

Fase 4 : Modellazione logica dell'ipertesto : il modello ADM

La modellazione logica dell'ipertesto rappresenta la parte centrale della metodologia Araneus;essa descrive come le pagine devono essere organizzate nel sito web:tale fase è specifica del Web ed ha lo scopo di dettagliare la struttura dellipertesto così come verrà mostrata dal browser.A questo livello viene ignorata l'implementazione fisica delle pagine e ci si concentra suul'atrazione degli elementi logici delle pagine,cioè le parti di informazione rilevanti nella pagina(come un testo o immagini) e la loro organizzazione(pagine flat,lista di pagine o pagine innestate).Per realizzare tale scopo è necessario trasformare lo schema concettuale NCM dell'ipertesto in uno schema logico,basato sul

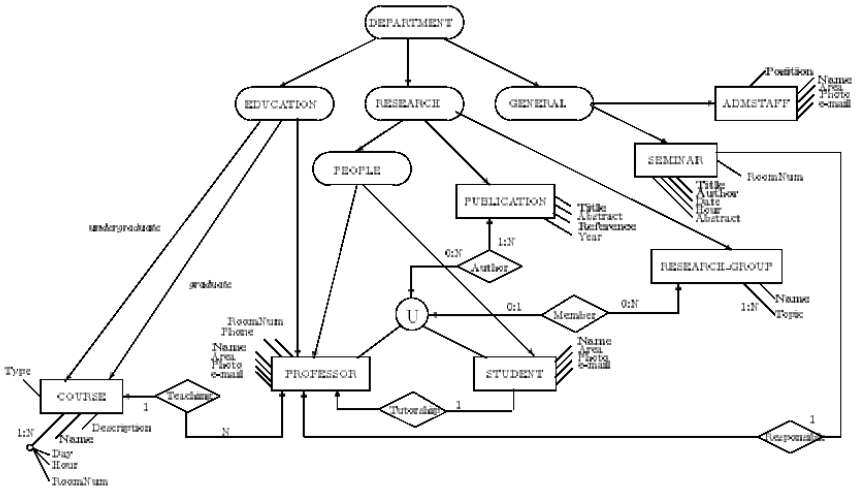

Figure 3.25: Esempio di schema NCM

modello *Araneus Data Model (ADM).*

Il modello *Araneus Data Model (ADM)* permette di descrivere in modo conciso la struttura delle pagine HTML astraendo i loro componenti logici.Tale modello ADM può essere definito "page-oriented",in quanto riconosce il ruolo centrale delle pagine nella struttura dell'ipertesto.
L'elemento fondamentale di ADM è lo *Schema di pagina*:uno schema di pagina ,la cui nozione ricorda quella di schema di relazione nei databases relazionali,è una descrizione intensionale di un insieme di pagine web con elementi in comune.Un'istanza di uno schema di pagina è una pagina Web,la quale viene considerata come un oggetto con un udentificatore(la URL associata) ed un insieme di attributi. È possibile definire uno schema di pagina come "unique" nel caso in cui abbia una sola istanza,cioè vi siano altre pagine con la stessa struttura.Tipicamente la home page di ogni sito appartiene a tale categoria.
Gli elementi di una pagina vengono descritti per mezzo di at-

tributi che possono essere di tipo semplice o complesso:

- *Attributi semplici*:hanno tipicamente un unico valore e corrispondono a parti atomiche dell'informazione come testi ,immagini o altri tipi multimediali.i *links* rappresentano attributi semplici speciali:ogni link è una coppia *(ancora,riferimento)*dove il *riferimento* è la URL della pagina di destinazione e l'*ancora* è un testo o un'immagine.Le *ancore* per i links possono essere stringhe costanti o corrispondere a tuple di attributi.

- *Attributi complessi*:sono attributi con più valori e rappresentano collezioni ordinate di oggetti,cioè liste di tuple.

ADM fornisce altri due costrutti particolarmente rilevanti nell'ambito Web:

- *Tipo di unione eterogenea*:viene definita un tipo di unione eterogenea ,come in NCM,allo scopo di rendere flessibile la modellazione coerentemente con la natura eterogenea del web.

- *Tipo di form*:le forms vengono utilizzate per eseguire programmi sul server e generare pagine dinamicamente,allo scopo di astrarre gli elementi logici di una form HTML questa viene vista come una *lista virtuale di tuple.*Ogni tupla ha tanti attributi quanti i campi fill-in della form e contiene un link alla pagina risultante.La lista di tali tuple è virtuale in quanto le tuple nom vengono memorizzate nella pagina ,ma corrispondono agli invii della form.

I costrutti ADM sono rappresentati graficamente in figura 3.26.
La trasformazione dallo schema concettuale NCM a quello logico ADM consiste nel mappare i nodi NCM negli schemi di pagina ADM e i links NCM in attributi di links ADM.In tale processo

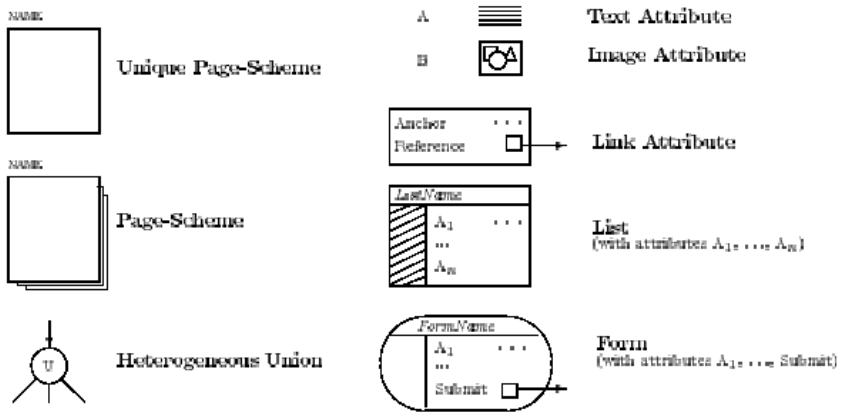

Figure 3.26: Rappresentazione grafica dei costrutti ADM

vi sono tre fasi metodologiche fondamentali che portano alla costruzione di un primo scheletro ADM :

- *Mapping delle macroentità*:le macroentità vengono mappate negli schemi di pagina;una macroentità puòessere mappata sia in uno schema di pagina non-unique che in uno schema unique con una lista.
  Vi sono molti modi di organizzare la presentazione di un'istanza di una macroentità utilizzando le pagine.Il modo più naturale è quello di usare una pagina per ogni istanza della macroentità.In alternativa possono essere usate più pagine collegate per rappresentare i diversi aspetti della stessa istanza oppure,in alcuni casi,tutte le istanze di una particolare macroentità possono essere presentate in una singola pagina.Nella conversione degli attributi ,gli attributi NCM a singolo valore vengono mappati negli attributi semplici ADM mentre gli attributi NCM a valori multipli sono mappati nelle liste ADM.Le figure 3.27 e 3.28 mostrano il mapping delle macroentità PROFESSORE e SEMINARIO nei

corrispondenti schemi di pagine:le istanze del primo vengono presentate in pagine separate,mentre le istanze del secondo sono presentate come una lista in una singola pagina.

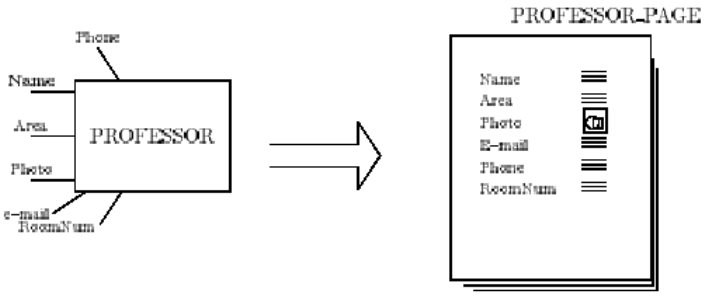

Figure 3.27: Esempio di mapping di macroentità in schemi di pagina

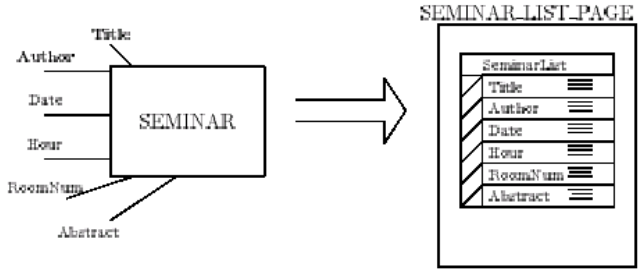

Figure 3.28: Esempio di mapping di macroentità in tuple

- *Mapping delle relazioni derivate*:le relazioni derivate tra le macroentità NCM vengono mappate in links tra i corrispondenti schemi di pagina.Come fase preliminare,tutte le relazioni NCM simmetriche vengono suddivise in due relzioni assimmetriche.In seguito,sono possibili scelte diverse riguardo al mapping in ADM delle macroentità partecipanti e alla cardinalità della relazione.Viene aggiunto un nuovo attributo allo schema di pagina sorgente(quello corrispondente alla macroentità sorgente):se la relazione è uno-

uno tale attributo è un attributo tipo link;se invece la relazione è uno-molti tale attributo è una lista di links.L'ancora del link in entrambi i casi deve essere scelta tr le chiavi descrittive della macroentità di destinazione;il riferimento del link è basato sulla URL dello schema di pagina di destinazione.Possono essere fatte scelte differenti se le istanze della macroentità di destinazione è stata mappata in pagina o in tuple di una lista o in uno schema di pagina:nel secondo caso è necessario inserire nella pagina un *offset*,cio è un puntatore ad un a porzione di pagina,allo scopo di fare riferimento ad una tupla spcifica nella lista.
La figura 3.29 mostra il mapping della relazione con uno-uno tra SEMINARIO e PROFESSORE:viene aggiunto un link alla LISTA-SEMINARIO nello schema di pagina corrispondente,in modo da permettere la navigazione dal seminario al corrispondente professore.Si pu ò inoltre notare che l'ancora di tale link corrisponde alla chiave decrittiva della macroentità di destinazione PROFESSORE(cioè l'attributo Nome).
In figura 3.30 la relazione uno-molti tra i PROFESSORI e le loro PUBBLICAZIONI viene suddivisa in due relazioni mappate separatamente:la relazione dai professori alle pubblicazioni è mappata in una lista di links nello schema di pagina PAGINA-PROFESSORE; ogni elemento della lista punta ad un elemento nella PAGINA-LISTA-PUBBLICAZIONI.

- *Mapping delle aggregazioni*:ogni nodo di aggregazione NCM viene mappato in uno schema di pagina ADM unique.I links di aggregazione NCM corrispondono agli attributi di link con alcune particolarità.Se il nodo di destinazione èun nodo di aggregazione o una macroentità mappata in uno schema di pagina unique viene usato un singolo link al corrispondente schema di pagina;al contrario,se la destinazione è un

nodo di unione o una macroentità mappata in uno schema di pagina,viene aggiunto un attributo di link ADM allo schema di pagina sorgente;ogni elemento della lista è un link ad un'istanza dello schema di pagina di destinazione.
Un esempio di mapping di aggregazioni è dato in figura 3.31 dove l'aggregazione RICERCA è mappata nello schema di pagina PAGINA-RICERCA,poichè uno dei componenti ,PERSONE,è un nodo di aggregazione,questo viene mappato nello schema di pagina PAGINA-PERSONE.

Dopo che tutte le primitive NCM sono state mappate nei costrutti ADM,basandosi sulle fasi descritte precedentemente,si ottiene un primo scheletro ADM.Anche se tale schema logico riflette lo schema concettuale NCM,può presentare delle limitazioni:i dati possono non essere sufficentemente organizzati e per accedere alle informazioni possono essere necessarie navigazioni comp-

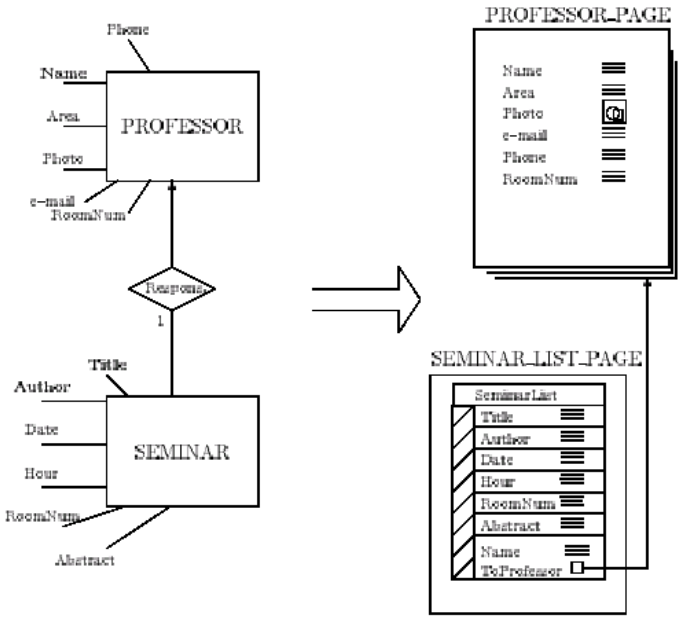

Figure 3.29: Esempio di mapping di relazioni direzionate

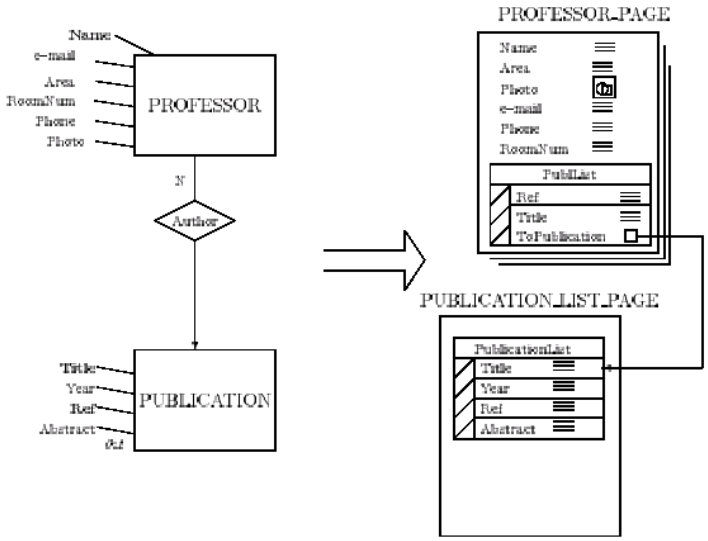

Figure 3.30: Esempio di mapping di relazioni direzionate

lesse.
Dunque molti schemi di pagina possono essere considerati troppo densi e poco effettivi.Allo scopo di risolvere tali problemi lo schema ADM può essere ristrutturato riorganizzando l'informazione ed i cammini di navigazionesenza modificare la struttura concettuale dell'ipertesto.
Tra le classi principali di riorganizzazione vi sono gli *Slicing page-schemes*.
Si tratta di schemi di pagina corrispondenti a macroentità troppo ricche che possono essere suddivisi in due (o più schemi di pagina,ogniuno dei quali presenta un sottoinsieme degli attributi dello schema di pagina originale;in questo modo si possono avere prospettive differenti dello stesso concetto.
Tale attività di *Slicing page-schemes* ricorda l'attività di *Slice design* di RMM.
Un'altra possibile trasformazione riguarda il caso di liste contenenti troppi elementi che rendono difficoltoso inserire dati all'interno

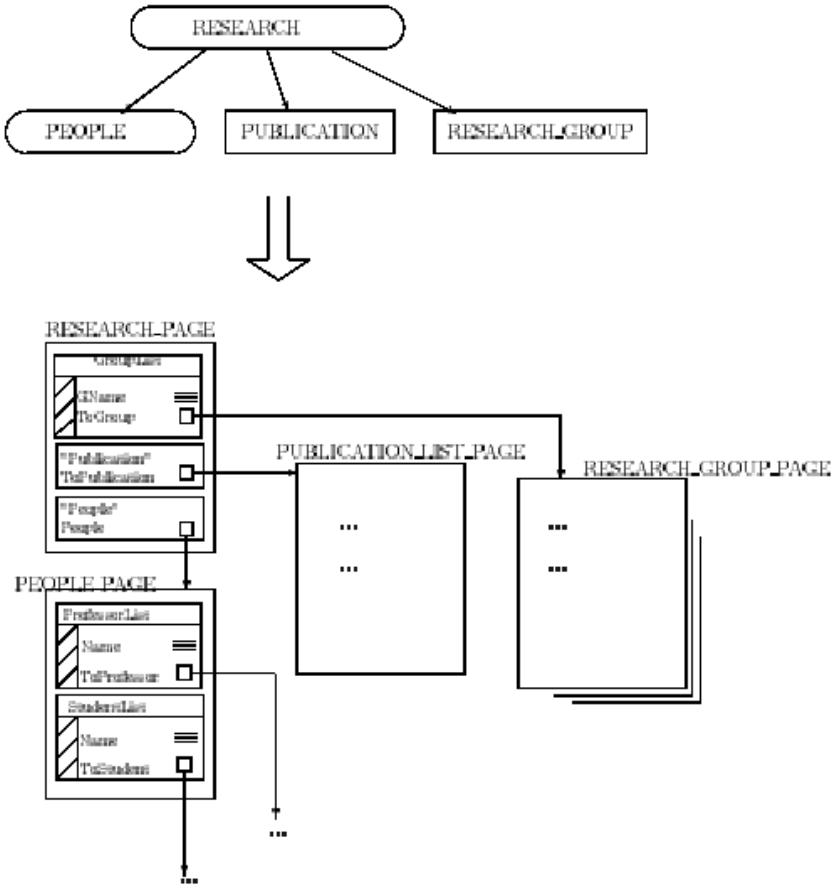

Figure 3.31: Esempio di mapping di aggregazioni

della pagina;in questo caso la lista può essere riorganizzata introducendo diversi livelli oppure può essere sostituita da una form.

Fase 5 : Modellazione della presentazione

Una volta sviluppato lo schema logico dell'ipertesto,per ogni schema di pagina deve essere modellato un layout;tale fase ,chiamata *modellazione della presentazione*,consiste nello scegliere il

modo giusto di visualizzare l0informazione all'interno di una pagina.La scelta del giusto layout di una pagina è molto importante per rendere la pagina interessante per l'utente e per trasportare meglio l'informazione.Come risultato della modellazione della presentazione,si produce un *template di pagina HTML*,cioè una singola pagina HTML prototipale,per ogni schema di pagina.Tali templates di pagina HTML vengono prodotti dal tool HOMER e possono essere basati su di uno stile specificato dallo sviluppatore.Poichè i templates di pagina sono file HTML ordinari è possibile raffinarli utilizzando un qualunque editor HTML e visualizzarli usando un browser ordinario. I templates raffinati vengono poi analizzati da HOMER,il quale associa ad ogni pagina da pubblicare nel sito i corrispondenti dettagli di presentazione.

Fase 6 : Mapping dell'ipertesto sul database e generazione delle pagine

La generazione delle pagine e quindi l'implementazione del sito web rappresenta la fase finale del processo di sviluppo.In tale fase si deve decidere:

- Il formato di destinazione(HTML o XML/XSL)
- Quali parti del sito devono essere materializzate in files e quali devono essere create dinamicamente utilizzando gli scripts.
- Quale linguaggio di script utilizzare(Java Server Page o Microsoft Active Server Page).

In seguito il sistema genera il codice sorgente dei programmi necessari per implementare il sito,in modo totalmente automatico.Tali programmi accedono al database sottostante,estraggono i dati,li organizzano sulla base dello schema ADM e li associano con

ogni parte di informazione del corrispondente stile di presentazione.Se il formato scelto è XML,il sistema genera anche i necessari fogli di stile XSL.L'automazione della generazione del codice sorgente è possibile poichè il sistema gestisce internamente una rappresentazione formale dei vari modelli e fasi di modellazione.In particolare ogni pagina dello schema ADM viene mappata nel database per mezzo di un'espressione di un'algebra relazionale innestata,chiamata *HOMER algebra*.In sostanza,la generazione delle pagine consiste nella valutazione della corrispondente espressione.

Analisi di Araneus sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali di Araneus descritti precedentemente permettono di verificare se tale metodologia soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,Araneus,come RMM riconosce tutti i tre livelli di contenuto,ipertesto e presentazione,ma si concentra solamente sui livelli di contenuto ed ipertesto.Una caratteristica unica di Araneus è che il livello di contenuto e di ipertesto vengono raffinati indipendentemente l'uno dall'altro.A livello di contenuto anche in Araneus,come in RMM,viene utilizzato il modello E/R.Considerando il livello di ipertesto,il modello concettuale *Navigation Conceptual Model (NCM)*viene raffinato dal modello logico *Araneus Data Model (ADM)*.Il livello di presentazione viene considerato nella modellazione della presentazione la quale consiste semplicemente nell'associare ad ogni schema di pagina ottenuto dal ADM un template di pagina HTML.Nella parte finale del processo ,dopo la modellazione della pre-

sentazione,viene effettuato il mapping tra l'ipertesto ed il contenuto informativo immagazzinato nel database.

- Dal punto di vista dei *aspetti*,analogamente a RMM,vengono considerati solamente gli aspetti strutturali per il livello di contenuto così come per quello di ipertesto.

- Dal punto di vista delle *fasi*,Araneus è caratterizzato da una chiara distinzione tra le fasi di modellazione concettuale e di modellazione logica a livello di contenuto e di ipertesto.

3.2.3 Strudel

Strudel [13] rappresenta,insieme ad Araneus,un altro grande progetto per lo sviluppo di siti web in base ad un approccio "model-driven".Il sistema Strudel applica i concetti dei DBMS al processo di creazione e gestione di siti web.In particolare,Strudel mostra il valore della specifica dichiarativa del contenuto e della struttura del sito e l'uso di query languages dichiarativi per definire automaticamente il sito.L'idea di base di Strudel è quella di separare la gestione dei dati del sito ,la gestione della struttura del sito e la presentazione grafica delle pagine HTML.
L'architettura del sistema Strudel è mostrata in figura 3.32 .
Il primo passo nello sviluppo di un sito web mediante Strudel è la definizione dei dati che dovranno essere disponibili nel sito.Tali dati possono essere memorizzati sia in sorgenti esterne (databases,files strutturati) o nel data-repository interno di Strudel.
Strudel si basa su un *modello dei dati semistrutturato* composto da grafi direzionati ed asociati ad etichette.Tale modello è stato introdotto per gestire dati semistrutturati i quali sono caratterizzati da pochi vincoli legati al tipo,da una struttura irregolare e da uno schema mancante o che cambia rapidamente.Tali dati semistrutturati facilitano l'integrazione di dati provenienti da sorgenti eterogenee,anche non-tradizionali.

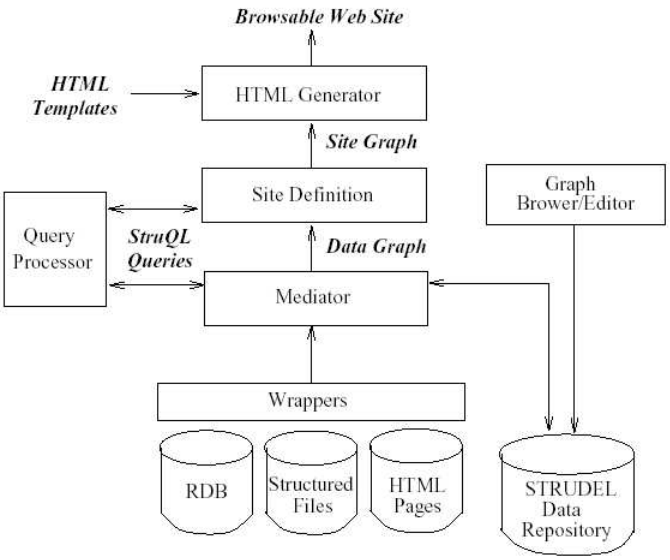

Figure 3.32: Architettura di Strudel

In tale modello dei dati semistrutturato,il database consiste di *oggetti* o *nodi*.Gli oggetti possono essere sia complessi,cioè un insieme di coppie attributo/oggetto) identificato da un identificatore unico,che valori atomici come interi, stringhe e files.Strudel supporta alcuni tipi atomici che compaiono abitualmente nelle pagine web, cioè URLs, PostScript, ASCII, immagini e files HTML.
Un insieme di *wrappers* specifici trasforma la rappresentazione esterna dei dati nel modello grafico descritto,ottenendo il *grafo dei dati*.
Per supportare la funzionalità di utilizzare dati provenienti da sorgenti eterogenee,Strudel include un *mediatore* il quale fornisce una vista uniforme dei dati sottostanti,indipendentemente da dove tali dati sono memorizzati.

Il secondo passo nello sviluppo del sito mediante Strudel consiste nella specifica dichiarativa della struttura del sito mediante

una *query di definizione del sito* utilizzando un particolare linguaggio di query tipico di Strudel chiamato *StruQL*.
StruQl è un linguaggio dichiarativo per effettuare queries e ristrutturare i grafi .Una query StruQL è composta da due clausole:

- Una clausola di selezione ,la quale specifica quale parte dei dati deve essere inclusa nel sito;

- Una clausola che specifica come manipolare l'insieme di oggetti ottenuti dalla selezione per creare un nuovo grafo dei dati,cioè un *grafo del sito*.

Per ottenere una presentazione grafica del sito pubblicabile dal browser,ad ogni oggetto del grafo del sito viene associato un template HTML.Il programmatore deve dunque scrivere i templates HTML i quali consistono in un insieme di codice HTML ed espressioni tipiche di Strudel che permettono l'accesso agli attributi di un certo oggetto.
Una volta ottenuto ogni oggetto con il proprio template HTML,il *generatore HTML* del sistema Strudel interpreta il template,sostituendo le espressioni specifiche con i valori HTML degli attributi dell'oggetto.Le pagine risultanti rappresentano il sito Web visualizzabile tramite browser.

Confronto tra i progetti Strudel ed Araneus

Strudel condivide con Araneus l'idea di separare i tre livelli di dati,struttura e presentazione nello sviluppo di siti web Data-Intensive;tuttavia vi sono alcune differenze tra Strudel ed HOMER,il CASE tool della metodologia Araneus.
Prima di tutto,HOMER si basa sulla tecnologia tradizionale dei DBMS relazionali (il modello E/R per il livello dei dati);al contrario ,la base di Strudel è un modello dei dati semistrutturati.
Strudel,proprio grazie a tale modello semistrutturato,risulta più

adatto all'integrazione di dati non-tradizionali(come testi e documenti) ma non può contare sull'efficenza e la robustezza dei databases relazionali.
Una seconda differenza importante è che in Strudel lo sviluppo di un sito Web richiede un'attività di scrittura di codice;al contrario,HOMER permette la produzione automatica di codice per l'implementazione del sito.

Analisi di Strudel sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali del progetto Strudel descritti precedentemente permettono di verificare se tale metodologia soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,Strudel,come Araneus ,riconosce tutti i tre livelli di contenuto,ipertesto e presentazione e ne sottolinea la chiara separazione.I livelli di contenuto ed ipertesto(struttura del sito) vengono rappresentati sulla base dello stesso modello semistrutturato ,rispettivamente attraverso il grafo dei dati ed il grafo del sito.A livello di presentazione in Strudel è richiesta una fase di programmazione per quanto riguarda i templates HTML associati ad ogni nodo del grafo del sito.

- Dal punto di vista dei *aspetti*,vengono considerati solamente gli aspetti strutturali per tutti i livelli.

- Dal punto di vista delle *fasi*,non è presente una chiara distinzione delle fasi di modellazione logica e concettuale.Il sistema Strudel copre tutte le fasi di sviluppo di un sito Web fino all'implementazione,tuttavia è caratterizzato da

una forte dipendenza dalla propria tecnologia prorietaria e da dettagli implementativi.

3.3 Linguaggi di modellazione web nell'ambito dell'*Object-oriented modelling*

3.3.1 OOHDM

L'*Object-Oriented Hypermedia Model*(*OOHDM*) [15] [16] [17] è un approccio "model-oriented" per lo sviluppo di applicazioni ipermediali.
OOHDM,come indica il nome,si basa sul modello HDM ma applica le tecniche dell'object-oriented ed in particolare fa uso dei diagrammi UML ormai considerati come standard per la modellazione in quanto la loro sintassi è chiara e ben conosciuta.

OOHDM prevede quattro attività distinte chiamate:

- Modellazione concettuale;
- Modellazione navigazionale;
- Modellazione dell'interfaccia astratta;
- Implementazione.

Durante ogniuna di tali attività viene costruito un insieme di modelli object-oriented che descrivono particolari concetti;tali modelli vengono ottenuti arricchendo le rappresentazioni ottenute nelle iterazioni precedenti.
La modellazione concettuale e quella navigazionale vengono separate in quanto riguardano concetti diversi legati alle applicazioni Web.Infatti,mentre la modellazione concettuale riflette gli oggetti e i comportamenti nel dominio applicativo,la modellazione navigazionale ha lo scopo di organizzare l'iperspazio tenendo conto dei profili utente .

La modellazione concettuale

Durante tale fase viene creato il modello del dominio dell'applicazione utilizzando primitive e principi della modellazione object-oriented simili a quelli dell'UML [2].Il prodotto di tale fase è un diagramma delle classi costituito da sottosistemi,classi e relazioni.Le differenze principali con UML sono l'uso di attributi multivalore e l'uso esplicito di direzioni nelle relazioni.Analogamente ad UML,le aggregazioni e le gerarchie di generalizzazione/specializzazione vengono utilizzate come meccanismi di astrazione.
Un esempio di attributo multivalore(chiamato *prospettiva*) può essere la descrizione di un oggetto o di una persona la quale può essere un testo o un'immagine.La sintassi per esprimere un'attributo multivalore è:

Descrizione[testo+,immagine]

dove il + indica che l'attributo descrizione ha una prospettiva di testo default ed una possibile prospettiva di immagine.
L'obiettivo della modellazione concettuale è quello di rappresentare il dominio applicativo nel modo più "naturale" possibile,cioè indipendente dai compiti dell'applicazione e dal tipo di utenti.In altre parole,la modellazione concettuale non dovrebbe riflettere il fatto che l'applicazione verrà implementata in ambiente Web,in quanto il modello che tiene conto di tale caratteristica dell'applicazione verrà sviluppato durante la modellazione navigazionale.
Le classi del modello concettuale verranno mappate nei nodi del modello navigazionale utilizzando un meccanismo di viste,mentre le relazioni verranno usate per definire i links tra i nodi.

La modellazione navigazionale

In OOHDM un'applicazione viene considerata come una vista navigazionale sul modello concettuale del dominio.OOHDM riconosce quindi che gli oggetti attraverso i quali l'utente naviga non sono oggetti concettuali ma un altro tipo di oggetti "costruiti" mediante un meccanismo di viste a partire da uno o più oggetti concettuali.È importante tuttavia sottolineare che la navigazione avviene attraverso links che possono non essere derivati direttamente dalle relazioni concettuali.
Per ogni profilo utente può essere definita una diversa struttura navigazionale che rifletta gli oggetti e le relazioni del modello concettuale sulla base delle azioni che tale tipo di utente deve poter compiere.
La struttura in classi della navigazione viene definita da uno scema contenenti classi di navigazione.In OOHDM è presente un insieme di tipi predefiniti di classi di navigazione:nodi,links,ancore e strutture d'accesso.Il significato di nodi,links ed ancore è quello usuale;le strutture d'accesso,come gli indici,rappresentano i possibili modi di iniziare la navigazione.
Applicazioni diverse basate sullo stesso dominio possono contenere diverse topologie di linking sulla base del profilo utente.
Ad esempio, nel sito di un Dipartimento possono essere definite una vista per studenti ed una per docenti e ricercatori.Nella seconda vista,il nodo di un professore conterrà informazioni sullo stipendio,le quali non saranno visualizzabili nella vista dello studente.
La definione degli oggetti navigazionali come viste sugli oggetti concettuali avviene mediante un linguaggio di definizione object-oriented che permette di copiare e/o filtrare attributi di classi concettuali (collegate tra loro) nella stessa classe rappresentante un Nodo e di creare classi rappresentanti Links selezionando appropriate relazioni.

Le classi Nodo sono definite utilizzando un linguaggio di query simile a quello in [26] .I nodi presentano attributi di un unico tipo,i link ancora, e pssono essere atomici o compositi.Le ancore sono istanze della classe Ancora.
I links connettono gli oggetti navigazionali,vengono definiti come viste sulle relazioni concettuali e possono essere uno-uno oppure uno-molti.La navigazione di un link può essere espressa definendo le semantiche navigazionali proceduralmente come risultato del comportamento del link oppure utilizzando una macchina di transizione di stato object-oriented simile agli StateCharts di UML.
Le strutture d'accesso (come indici e cammini guidati) rappresentano modi alternativi per la navigazione all'interno dell'applicazione e sono anch'esse definite come classi.
A volte si ha la necessità di astrazioni di livello più alto per definre strutture navigazionali più ricche di significati ed usi;a tale scopo vengon introdotti nodi e links che non corrispondono ad entità o relazioni concettuali .

Le applicazioni Web solitamente contengono collezioni di pagine con concetti simili;tali collezioni possono essere esplorate in modi diversi sulla base delle azioni che l'utente vuole realizzare.Ad esempio,in un negozio di libri elettronico,l'utente può voler esplorare i libri di un certo autore piuttosto che i libri di un certo movimento letterario.Come risultato dell'organizzazione degli oggetti navigazionali in gruppi,molte operazioni fanno riferimento ad una navigazione intra-gruppo;ad esempio "precedente" o "successivo".Perciò i gruppi di oggetti definiscono links che vriguardano tale navigazione e che dunque non hanno diretta controparte nel modello concettuale.
OOHDM struttura dunque lo spazio navigazionale in gruppi ,chiamati*contesti navigazionali* e rappresentati nello *schema dei contesti*.Ogni contesto navigazionale è un insieme di nodi e viene

descritto indicando la sua struttura navigazionale interna (ad esempio se vi si può accedere sequenzialmente),un punto d'accesso e gli indici associati.
In OOHDM vi sono quattro casi speciali di contesti navigazionali:

- *Contesto derivato da una classe semplice*:include tutti gli oggetti di una classe che soddisfa una determinata proprietà sugli attributi:ad esempio,"quadri con pittore=Van gogh" o "professori con ruolo=associato".

- *Contesto derivato da una classe* : è un insieme di contesti derivati da una classe semplice,dove la proprietà di definizione di ogni contesto è parametrizzata:ad esempio ,"quadri in base al pittore" o "professori in base al ruolo".

- *Contesto derivato da un link semplice*:include tutti gli oggetti n relazione con un dato oggetto:ad esempio "corsi tenuti dal prof.Smith".

- *Contesto derivato da un link*:è un insieme di contesti derivati da un link semplice,ogniuno dei quali viene ottenuto variando l'elemento di partenza del link:ad esempio "corsi tenuti da un professore" o "quadri di un pittore" dove professore e pittore possono variare.

I contesti possono variare durante la navigazione,sia perchè l'utente può creare o modificare gli elementi dell'informazione(elementi navigazionali) andando quindi a modificare il conteso derivato,sia perchè lgi utenti possono esplicitamente inserire o togliere oggetti dal contesto.Tali contesti sono detti *contesti dinamici* ed un esempio può essere il carrello della spesa.
Quando lo stesso nodo può apparire in più di un contesto risulta necessario esprimere le peculiarità di tale nodo all'interno di ogni particolare contesto.

In OOHDM per ogni nodo e per ogni contesto in cui esso appare,viene definita una *classe NelContesto* che funge da *decoratore* per i nodi quando vi si accede in un particolare contesto.
I decoratori rappresentano un'alternativa alle sottoclassi e permettono di evitare la definizione di molte sottoclassi di una classe Nodo di base.
Le *classi NelContesto* sono organizzate in gerarchie con alcune classi base e non sono mappate dallo schema concettuale in quanto rappresentano un concetto puramente navigazionale.

La modellazione dell'interfaccia astratta

Nella fase di modellazione dell'interfaccia astratta si specifica da quali oggetti deve essere composta l'interfaccia dell'applicazione verso l'utente e quale deve essere il comportamento di tale interfaccia.Per ogni attributo di un nodo (ancore comprese)si deve definire come dovrà apparire.Distinguendo tra modellazione della navigazione e dell'interfaccia è possibile costruire interfacce diverse per la stessa applicazione.
Durante tale attività può essere definito il modo in cui dovranno apparire oggetti navigazionali diversi,quali oggetti permettono la navigazione ed il modo in cui gli oggetti multimediali dell'interfaccia saranno sincronizzati.
in OOHDM viene utilizzato un approccio *Abstract Data View (ADV)* per descriver l'interfaccia utente delle applicazioni Web.
La costruzione di un modello formale dell'interfaccia è un'attività molto importante in quanto le interfacce utente tendono a cambiare più rapidamente delle topologie di navigazione.

L'implementazione

Durante l'attività di implementazione gli oggetti concettuali,navigazionali e dell'interfaccia vengono mappati nel particolare ambiente d'esecuzione di destinazione.

Quando tale ambiente non è completamente object-oriented è neccesario effettuare il mapping degli oggetti concettuali,navigazionali e dell'interfaccia in oggetti concreti ,cioè quelli disponibili nell'ambiente di implementazione.Ciò può comportare la definizione di pagine HTML,di scripts in qualche linguaggio e di queries al database. OOHDM non presuppone una particolare strategia di implementazione;tuttavia,tenendo conto dell'approccio object-oriented seguito,l'implementazione di complesse strutture navigazionali in applicazioni Web mediante la tecnologia oject-oriented risulta molto flessibile,ad esempio utilizzando Java.

Analisi di OOHDM sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali dell' *Object-Oriented Hypermedia Model* descritti precedentemente permettono di verificare se tale metodologia soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,OOHDM separa chiaramente i tre livelli delle applicazioni Web.A livello di contenuto OOHDM utilizza la tecnica di modellazione UML[2] come formalismo per modellare struttura e comportamento.Il livello di ipertesto viene modellato utilizzando due concetti differenti:un diagramma delle classi UML per definire le classi navigazionali,come descritto precedentemente,ed uno schema dei contesti per modellare le strutture d'accesso alle classi navigazionali.A livello di presentazione un altro formalismo descritto in precedenza,l'ADM,viene usato per descrivere la struttura di layout degli oggetti di navigazione.

- Dal punto di vista dei *aspetti*,OOHDM tiene maggiormente in considerazione l'aspetto comportamentale rispetto agli altri linguaggi descritti finora.L'aspetto comportamentale

viene trattato maggiormente nei livelli di ipertesto e di presentazione modellando,rispettivamente, l'attivazione/disattivazione degli oggetti durante la navigazione e la reazione ad eventi esterni provocati dall'utente(utilizzando i cosidetti ADV-Charts,una derivazione degli statecharts).

- Dal punto di vista delle *fasi*,OOHDM distingue principalmente tra tre fasi che in parte corrispondono alla dimensione dei livelli:modellazione concettuale,modellazione navigazionale,modellazione dell'interfaccia astratta.

3.3.2 OO-H Method

L' *Object-oriented Hypermedia (OO-H) Method* ,seguendo l'approccio di OOHDM [15] e di Conallen [19],vede i sistemi Web come artefatti software unificati dove la struttura,il comportamento e la presentazione sono gli elementi di base che devono essere propriamente combinati per ottenere un software finale corretto.Il metodo OO-H è un modello generico che fornisce una notazione per sviluppare interfacce Web e connetterle con i preesistenti moduli della logica dell'applicazione.
Il metodo OO-H si basa sugli approcci tradizionali nell'ambito della modellazione concettuali fondati su UML.In particolare ,il metodo OO-H viene considerato come un'estensione del metodo OO [27],anch'esso basato su UML.
Tale metodo può essere considerato (figura 3.33) come un'ambiente automatico di produzione software i cui componenti principali sono:

- Un insieme di viste per definire la struttura del sistema (punto di vista statico) ed il comportamento del sistema (punto di vista dinamico).

- Un compilatore del modello che genera le sorgenti di dati ed i moduli logici nell'ambiente di implementazione desiderato

Il modello OO-H estende tali viste con due nuovi diagrammi complementari:

- Un *Navigational Access Diagram (NAD)* ,il quale definisce una vista di navigazione.
- Un *Abstract Presentation Diagram (APD)* ,il quale raccoglie i concetti relativi alla presentazione.

Sia NAD che APD modellano l'informazione relativa all'interfaccia web con l'aiuto di un insieme di patterns,definiti in un *Catalogo dei Patterns di Interfaccia* integrato nel metodo OO-H.
Analogamente al metodo OO,il metodo OO-H fornisce un compilatore del modello che genera il front-end dell'applicazione web per la piattaforma client ed il linguaggio client(HTML,XML) desiderati.
L'approccio del metodo OO-H risulta dunque una vera soluzione Internet a tre-livelli,come rappresentato in figura 3.33.

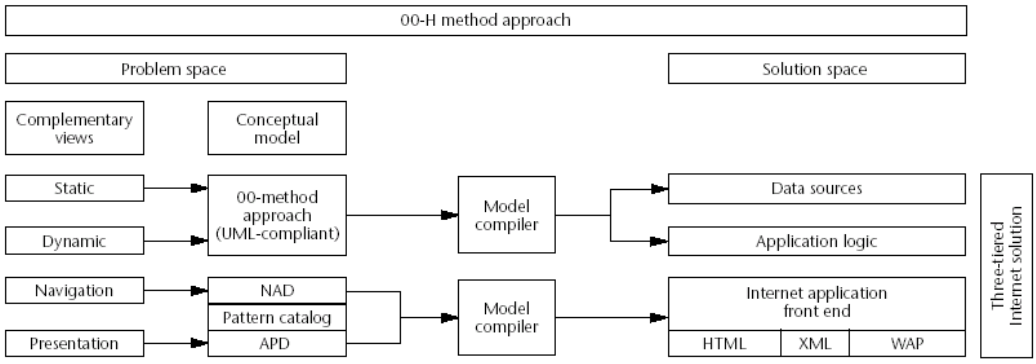

Figure 3.33: Il metodo OO-H

Il metodo OO-H include dunque il seguente insieme di notazioni,tecniche e strumenti che permettono la modellazione di interfacce web:

- Un processo di design;

- Un catalogo di patterns;
- Un NAD;
- Un APD;
- Un CASE tool che automatizza lo sviluppo delle applicazioni Web modellate con il metodo OO-H.

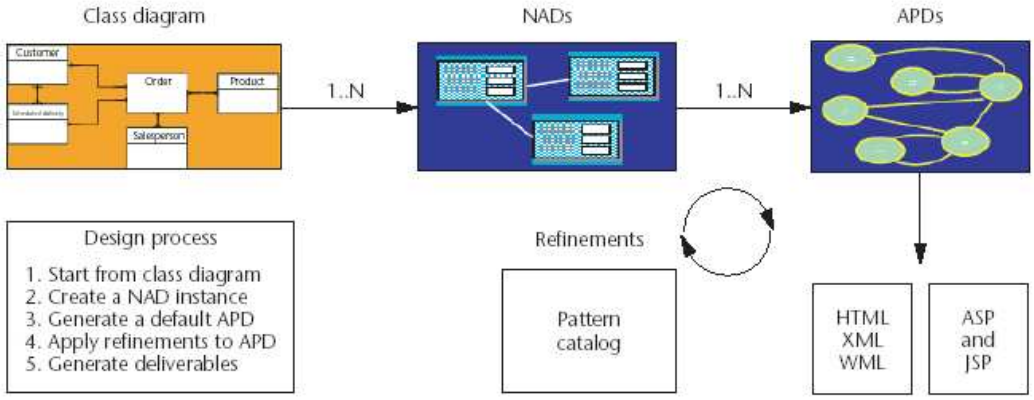

Figure 3.34: Metodo OO-H:il processo di design

Il processo di design

Il processo di design definisce le fasi da seguire per la costruzione di un'interfaccia Web funzionale che presenti i requisiti richiesti dal cliente.
Come mostrato in figura 3.34 ,il processo di design del metodo OO-H inizia con la modellazione della struttura del dominio informativo mediante un diagramma delle classi UML.In seguito vengono modellate (1..N) istanze NAD personalizzate,una per ogni tipo di utente.
Ogni istanza NAD rappresenta le informazioni,i servizi ed i cammini navigazionali per i requisiti di navigazione specificati dall'utente

associato.Una volta costruiti i diagrammi NADs,può essere generata un'interfaccia web di tipo prototipale.Tuttavia,le implementazioni finali solitamente risultano più sofisticate,sia dal punto di vista grafico che dell'usabilità.
Per migliorare la qualità dell'interfaccia,il metodo OO-H introduce un secondo diagramma,l'APD,basato sul concetto di *template*,il quale deriva la sua struttura di default direttamente dal NAD.
Possono essere definiti diversi (1..N)APDs per lo stesso NAD per indicare modi diversi di visualizzare gli stessi requisiti legati alla modellazione.
Allo scopo di aiutare il designer a raffinare tale struttura mantenendone la qualità,nel metodo OO-H è presente un catalogo dei patterns contenente un insieme di costrutti che effettivamente risolvono alcuni problemi.Tale approccio facilita l'uso di esperienze nel campo del design e la consistenza tra i moduli dell'interfaccia e tra le interfaccie delle applicazioni.
Una volta raffinato il APD,viene generato il front-end dell'applicazione Web,statico o dinamico,per l'ambiente desiderato(come HTML,XML,Active Server Pages(ASP) o JavaServer Pages (JSP)).

Il catalogo dei patterns

Il catalogo dei patterns del metodo OO-H fornisce un linguaggio dei patterns per l'interfaccia ipermediale.Tale linguaggio può essere visto come una collezione parzilamente ordinata di patterns in relazione tra loro.IL catalogo contiene una serie di patterns astratti con alcuni possibili modi di implementazione.Ogni pattern ha una delle sue possibili imlementazioni considerata come default.Inoltre ogni implementazione ha associata una *regola di trasformazione* che guida le trasformazioni nel diagramma quando viene applicato il pattern.Tali trasformazioni riguardano la creazione di nuove pagine,la ridirezione dei links tra le pagine

o la creazione di nuove dipendenze tra le pagine.
Una delle caratteristiche principali del catalogo dei patterns del metodo OO-H è quella di essere "centrato sull'utente";infatti la granularità a cui vengono descritti i patterns fornisce al designer nuovi meccanismi per soddisfare completamente i requisiti specificati dall'utente.
I patterns presenti nel catalogo forniscono soluzioni alternative ai tradizionali problemi legati all'ipermediale,considerati dal punto di vista dell'utente.

Il Navigational Access Diagram (NAD)

Il modello di navigazione nel metodo OO-H è formato da più diagrammi NAD,almeno uno per ogni tipo di utente la fine di rappresentarne la corrispondente vista del sistema.
Un NAD è basato su quattro tipi di costrutti:

- *Classi di navigazione* :sono classi del dominio applicativo arricchite i cui attributi sono stati ristretti sulla base dei permessi d'accesso dell'utente e dei requisiti di navigazione specificati.Un semplice arricchimento è la differenziazione tra tre tipi di attributi: *V-attributi* ,attributi visibili, *R-attributi* ,attributi referenziati che vengono visualizzati dopo una richiesta dell'utente, ed *H-attributi* ,attributi nascosti che vengono mostrati solo quando ènecessaria una visione globale ed esaustiva del sistema.

- *Targets di navigazione* :sono gruppi di elementi del modello che collaborano alla soddisfazione di ogni requisito navigazionale specificato dall'utente.

- *Links di navigazione* :definiscono i cammini di navigazione che l'utente può seguire.
  Tali links possono avere associato un pattern di navigazione o un insieme di filtri di navigazione,i quali forniscono

ulteriori informazioni per la costruzione del modello navigazionale.
Si possono distinguere quattro tipi di link: i *I-links (Internal links)*,links che definiscono la navigazione all'interno di un dato target navigazionale,i *T-links (Traversal links)*,links definiti tra le classi di navigazione appartenenti a targets diversi,i *R-links (Requirement links)*,links che puntano al punto di inizio della navigazione di ogni target navigazionale ed infine i *S-links (Service links)*che mostrano i servizi disponibili per il tipo di utente associato a quel NAD.

- *Collezioni*:rappresentano le possibili strutture gerarchiche definite sulle classi o sui targets navigazionali.Esse mettono a disposizione dell'utente nuovi modi di accedere all'informazione. Il metodo OO-H alcuni tipi di collezioni tra cui: le *C-collezioni (Classifier collections)*,dove il criterio di selezione della popolazione di oggetti è predefinito,e le *S-collezioni (Selector collections)* nelle quali l'utente può introdurre il criterio a tempo di esecuzione.

L' Abstract Presentation Diagram (APD)

Nel metodo OO-H per la specifica dell'aspetto grafico e della struttura di pagina del Web viene addottato un approccio basato sui *templates* .Allo scopo di separare i vari aspetti che contribuiscono all'aspetto ed al comportamento dell'interfaccia finale,vengono definiti cinque tipi di templates,espressi come documenti XML.Ad ogni documento ,cioè ad ogni tipo di template, viene associata una *document type definition (DTD)* per poter definire le etichette e la struttura del documento.
I cinque tipi principali di templates definiti nel metodo OO-H sono:

- tStruct: le istanze di tale tipo di template definiscono l'informazione che deve comparire sulla pagina astratta.

- tStyle: le istanze di tale tipo di template definiscono caratteristiche come il posizionamento fisico degli elementi e la tipografia.

- tForm: le istanze di tale tipo di template definiscono i dati richiesti all'utente per poter interagire con il sistema.

- tFunction: le istanze di tale tipo di template svolgono le funzionalità del client;sono basate su una specifica *document object model (OMD)*(vedi http://www.w3.org/XML/),la quale si propone di fornire una piattaforma ed un'interfaccia indipendente dal linguaggio alla struttura ed al contenuto di documenti XML e HTML.Le funzioni definite in questo tipo di template sono indipendetni dal linguaggio ,dunque devono essere mappate in un linguaggio di destinazione(JavaScript o altri) per diventare operative.

- tWindow: le istanze di tale tipo di template definiscono un insieme di viste simultanee disponibili per l'utente.

Diagramma APD di default:
Il metodo OO-H definisce un insieme di regole per il mapping dei diversi elementi contenuti nel diagramma NAD in elementi equivalenti del diagramma APD.Tali regole sono:

- *V-attributi*,*I-links*, *T-links* , *R-links*: compaiono come elementi della pagina tStruct.

- *C-collezioni*, *S-collezioni*:sono alberi statici e nel diagramma APD sono definite per mezzo di una singola pagina astratta tStruct la quale contiene una struttura tipo-albero formata da elementi di link che puntano ad altre tStruct.Una *S-collezione* può anche comportare la creazione di un nuovo template tForm,se il filtro ad essa associato rigurda un insieme di valori dipendenti dinamicamente dall'utente.

- *T-collezioni*, *S-links*:possono generare una pagina astratta tForm iniziale,contente i parametri che l'utente deve introdurre per tutti i comandi coinvolti nella transazione.Tali collezioni e links devono contenere un elemento di link che punti ad un comando(considerato come un metodo o una transazione,nel metodo OO-H).Se il comando ritorna qualcosa,verrà generata un'altra pagina tStruct con la stuttura definita da tale valore di ritorno(un oggetto o un'insieme di oggetti).

- *R-attributi*:causano la creazione di una nuova pagina astratta tStruct nel diagramma e di un nuovo link che punta alla pagina originale.

- *Patterns di navigazione NAD*:tutti gli indici ed i cammini guidati vengono mappati in APD per mezzo di un insieme di links definiti su una singola pagina tStruct.

Il diagramma APD di default fornisce un'interfaccia funzionale ma anche semplice,la quale tuttavia richiede ulteriori raffinamenti per poter essere utilizzabile nell'applicazione finale.Essa può comunque essere utilizzata come un prototipo con il quale verificare che i requisiti specificati dall'utente sia stati soddisfatti .

Raffinamento del diagramma APD:
Il processo di raffinamento modifica il diagramma APD di default.Tale processo consiste nell'applicare una serie di patterns collegati all'APD contenuti nel catalogo.La scelta del designer di utilizzare o meno tali patterns influenza il diagramma APD finale ,sia nella struttura che nel contenuto.Nel metodo OO-H è presente una regola di trasformazione che descrive i cambiamenti introdotti nell'APD in seguito all'applicazione di un pattern associato ad una qualunque delle sue implementazioni.

Ogni APD ha un nome identificatore che ha il ruolo di elemento di root dal quale derivano tutti gli altri elementi del diagramma;tale elemento di root è del tipo APDSchema.Le pagine vengo aggiunte allo schema attraverso il servizio "addAPDPage" associato all'oggetto APDSchema.Ogni volta che si aggiunge una pagina allo schema si devono definirne il nome ed il tipo;inoltre le nuove pagine possono avere associato un qualunque numero di funzioni ,le quali devono essere precedentemente definite in un function repository di APD.
In generale,l'utilizzo di patterns può provocare la creazione o la modifica di un qualunque tipo di pagina astratta.
Nel metodo OO-H si può anche scegliere se rendere visibili o meno i patterns scelti.Ad esempio,l'applicazione di un pattern di errore può causare la creazione di una nuova pagina tStruct nello schema,ma non contiene informazioni con significati importanti e dunque si può scegliere di non mostrare tale patterns nel diagramma.

L' implementazione dell'ADP

Dopo aver raffinato l'APD,gli elementi dell'interfaccia modellata vengono forniti ad uno strumento compilatore del modello il quale genera l'interfaccia Web operativa sulla base della conoscenza dell'ambiente di destinazione.Tale compilatore del modello è inserito nel CASE tool del metodo OO-H,il quale fornisce un'interfaccia per l'eleborazione di tutti i modelli OO-H ma sopratutto ,permette di generare il front-end dell'applicazione Web in ambienti industriali di sviluppo del software ben conosciuti.
È stata sviluppata un'applicazione campione utilizzando Java Server Pages e componenti Java Beans come tecnologia server ed HTML come tecnologia client,tuttavia il CASE tool permette di generare codice anche per altri ambienti(ad esempio dispositivi mobili WAP).

Analisi del metodo OO-H sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali del metodo OO-H descritti precedentemente permettono di verificare se tale metodo soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,il metodo OO-H distingue chiaramente tra i tre livelli di struttura,ipertesto ,navigazione.A livello di struttura vengono utilizzati il diagrammi delle classi UML,mentre ai livelli di ipertesto e presentazione vengono utilizzati due nuovi diagrammi:

  - Il *Navigational Access Diagram (NAD)*
  - Il *Abstract Presentation Diagram (APD)*

  A livello di ipertesto sono presenti più diagrammi NAD ognuno associato ad ogni tipo di utente,similmente alle *siteviews*definite in WebML.
  Analogamente ,a livello di presentazione,ad ogni NAD possono corrispondere più APDs per permettere di visualizzare la stessa informazione in modi diversi.Per quanto rigurda il mapping tra i livelli,viene particolarmente trattato il mapping tra i livelli di pertesto e presentazione:vengono definite delle regole di mapping tra il diagramma NAD ed il corrispondente APD di default,le quali rendono tale mapping chiaro ma scarsamente flessibile.

- Dal punto di vista dei *aspetti*,ai vari livelli i diagrammi vengono considerati come viste per rappresentare il sistema sia dal punto di vista strutturale(statico) che dal punto di vista comportamentale(dinamico).

- Dal punto di vista delle *fasi*,il metodo OO-H definisce un processo di design di interfacce Web completo(dalla model-

lazione del contenuto all'implementazione)ad un alto livello di astrazione,similmente agli altri approcci basati sulla modellazione concettuale.La caratteristica del metodo OO-H è tuttavia quella di descrivere un'interfaccia Web indipendente dai dispositivi utilizzati dal client perla visualizzazione dell'informazione ed in generale dall'ambiente di sviluppo desiderato.

3.3.3 UML esteso di Conallen

L'approccio di *James Conallen* [19] [20] rappresenta la proposta più nota di estensione della notazione UML allo scopo di adattarla alla specifica di applicazioni Web.Tale approccio risulta completamente differente rispetto alle proposte descritte fin ora,proprio perchè totalmente proteso verso l'estensione di una metodologia standard come l'UML.
Gli stessi creatori di UML ammettono che vi sono situazioni nelle quali UML può non essere sufficente per descrivere tutti i significati rilevanti di un certo dominio;uno di questi domini è il Web.Tuttavia la proposta di Conallen ,anche chiamata *UML-esteso*, si basa sul fatto che UML è una notazione adattabile a qualsiasi dominio applicativo,grazie alla possibilità di estendere il suo nucleo centrale di concetti (package,oggetti,classi ed associazioni) con la definizione di:

- *Stereotipi*:si tratta di versioni particolari dei concetti base che definiscono un nuovo significato semantico per gli elementi della modellazione utile per descrivere un particolare settore applicativo.

- *Valori etichettati*:sono coppie di valori chiave che possono essere associate ad un elemento della modellazione .

- *Vincoli*:sono regole che definiscono la forma corretta del modello.Esse possono essere espresse sia con un testo libero

che con il pi ù formale Object Constraint Language(OCL).

Il dominio applicativo al quale Conallen si propone di estendere la notazione UML è quello delle applicazioni Web ,la cui architettura è descritta nel capitolo 1 nella sezione 1.3.2.In tale ambito applicativo Conallen si propone modellare,mantenendo il livello corretto di astrazione,non tanto i dettagli del Web Server o del Browser quanto le pagine,i links tra esse ,il loro contenuto dinamico e tutti i meccanismi tecnologici che compongono il front-end di un'applicazione Web(vedi1.3.2).
Individuati gli elementi da modellare,essi devono essere mappati sugli elementi di modellazione UML.Il mapping più naturale risulta essere quello delle pagine in classi UML,dei links in associazioni e degli script delle pagine in operazioni della classe UML.Tale astrazione risulta errata nel momento in cui si considera che una pagina web può contenere un insieme di scripts che si eseguono nel server(preparazione del contenuto dinamico delle pagine) e un insieme di scripts completamente diverso che si esegue nel client(ad es.,JavaScript).In tale contesto,osservando una classe del modello che rappresenta una pagina web si ha confusione su quali operazioni,attributi e relazioni sono attivi mentre la pagina viene preparata dal sever o quando l'utente sta interagendo con la pagina dal client.
La soluzione migliore per tale problema è il principio della *"separation of concerns"*,cioè la considerazione del fatto che il comportamento di una pagina web nel server è completamente diverso da quello nel client.
Applicando tale principio l'aspetto dal lato server di una pagina web può essere modellato con una classe,l'aspetto dal lato client con un'altra.
È possibile distinguere tra le due classi utilizzando un meccanismo di estensione UML allo scopo di definire degli stereotipi per ogni *server page* e *client page*.Le classi stereotipate vengono

rappresentate nel diagramma delle classi UML associate con il proprio nome stereotipato posto tra particolari virgolette ,come si vede in figura 3.35.Per le pagine web,gli stereotipi indicano che la classe è un'astrazione di un comportamento logico della pagina nel client o nel server.
Le due astrazioni sono collegate da una relazione direzionale tra esse.Tale associazione è stereotipata come *build*,in quanto si può affermare che una pagina web "costruisce" una pagina client (figura3.35).Infatti ogni pagina web dinamica (il cui contenuto viene determinato a runtime) è costruita con una pagina server;ogni pagina client è costruita dau na sola pagina server,mentre una pagina server può determinare più pagine client.

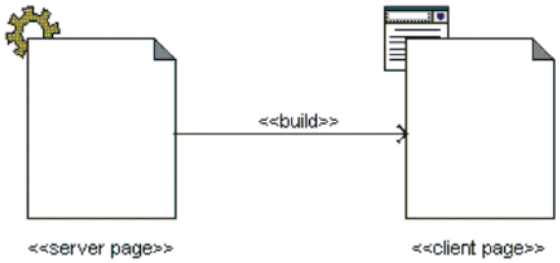

Figure 3.35: Associazione *build* tra pagina sever e pagina client

I links che rappresentano i possibili cammini navigazionali,cioè le relazioni tra pagine Web,vengono espressi nel modello con un'associazione stereotipata *link*.I links sono implementati nel sistema come la richiesta di una pagina web,dunque un'associazione ¡¡link¿¿ con una pagina client equivale,nella maggiorparte dei casi,ad un'associazione *link* con la pagina server che costruisce tale pagina client.
Tale associazione ha sempre origine da una pagina client e punta ad un'altra pagina client o ad una pagina server.
Per definire i parametri passati attraverso un link(richiesta) ven-

gono utilizzati i valori etichettati.Il valore etichettato *Parameters* dell'associazione *link* è una lista di nomi di parametri (e valori opzionali) che il server si aspetta ed utilizza per processare la richiesta.
In figura 3.36,la pagina SearchResults contenente i risultati della ricerca contiene un numero variabile di links (0..*) verso la pagina server GetProcuct; ogni link ha un valore diverso del parametro productId.
La pagina GetProduct costruisce la pagina ProductDetails con i dettagli del prodotto specificato dal parametro.

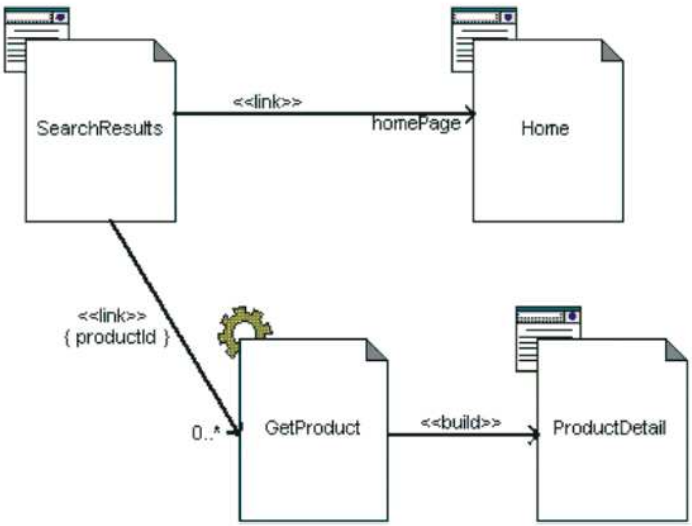

Figure 3.36: Uso dei parametri di link

Utilizzando gli stereotipi,risulta più facile modellare gli script e le relazioni di una pagina.Le operazioni della classe server page diventano funzioni negli scripts lato server della pagina ed i suoi attributi diventano variabili globalmente accessibili dalle funzioni della pagina.Analogamente le operazioni e gli attributi della classe client page diventano funzioni e variabili visibili nel client.Il vantaggio principale della separazione tra gli aspetti lato

client e lato server di una pagina in due classi diverse è nelle relazioni tra le pagine e le altre classi del sistema.Le pagine client sono modellate con relazioni con le risorse lato client:Java Applets,ActiveX controls,plug-ins(figura3.37).Le pagine server sono modellate con le relazioni con le risorse lato server:componenti del middle tier(vedi1.3.1),componenti di accesso al database(figura3.38).

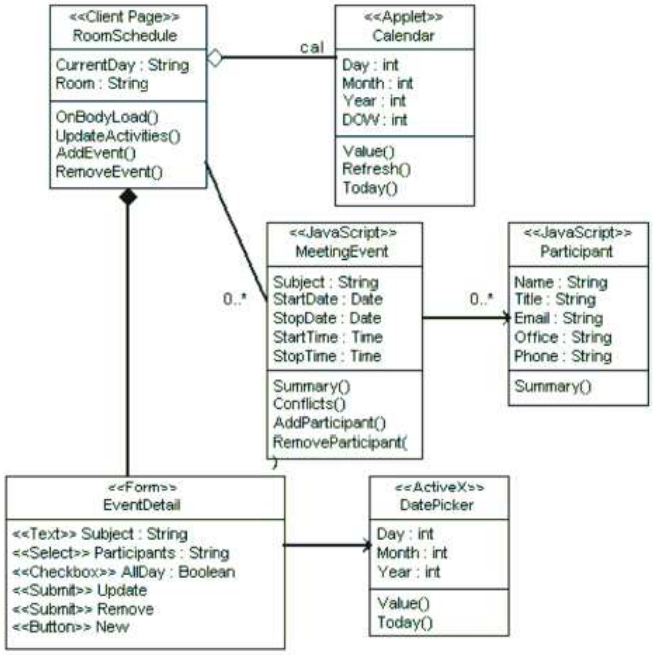

Figure 3.37: Collaborazioni del client

Uno dei maggiori vantaggi dell'utilizzo di stereotipi per modellare i comportamenti logici delle pagine web è che le loro collaborazioni con i componenti lato server possono essere espressi nello stesso modo di qualunque altra ccollaborazione lato server.La pagina server è semplicemente un'altra classe che partecipa alla logica di business del sistema.Ad un livello più concettuale,le pagine server hanno il ruolo di controllori che organizzano l'attività degli oggetti di usiness necessaria per adempiere all'obiettivo

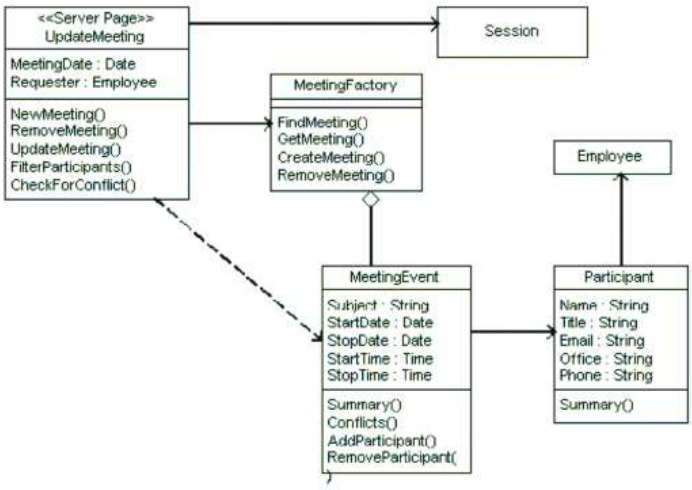

Figure 3.38: Collaborazioni del server

lanciato dalla richiesta del browser.
Dal lato client le collaborazioni possono essere più complicate,anche a causa della variet à di tecnologie coinvolte.Una pagina client è,nella sua forma più semplice,un documento HTML contenente informazioni sia sul contenuto che sulla presentazione.

Le Forms:
Le forms rappresentano il principale meccanismo di data-entry delle pagine web;esse sono definite come un documento HTML con un'etichetta *form.*
Ogni form specifica la pagina alla quale deve essere sottomessa;inoltre contiene elementi di input ,tutti espressi come etichette HTML.
Le etichette più comuni sono :*input*,*select* e *textarea.*L'etichetta di input può essere un campo di testo,un bottone o un'immagine.
La modellazione delle forms implica un'altra classe stereotipata,*form.*Una form non ha operazioni poichè ogni operazione che può essere definita in un'etichetta *form* in realtà appartiene alla classe client.Ogni form ha inoltre una relazione ,stereotipata con *sub-*

mit,con la pagina server la quale processa la sottomissione della form.
Poichè le forms sono completamente contenute in un documento HTML,sono espresse con una forte aggregazione nel diagramma delle classi UML.
La figura 3.39 mostra una semplice pagina del carrello della spesa che definisce una form e mostra la relazione di submit alla pagina server.

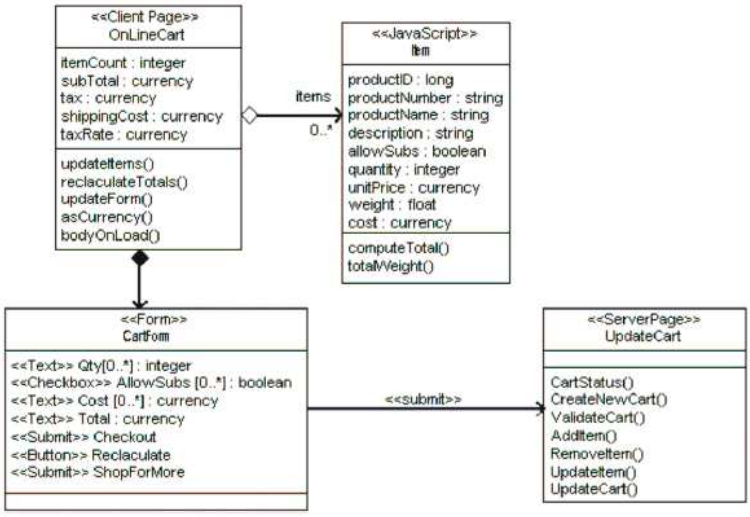

Figure 3.39: Modellazione delle forms

Nella figura 3.39 ,la classe stereotipata ¡¡JavaScript¿¿ è un oggetto che rappresenta i prodotti nel carrello della spesa.Nella descrizione delle proprietà della form che hanno un numero variabile di istanze viene utilizzata la sintassi di array.
Poichè tutta l'attività nella pagina client è eseguita in JavaScript,il quale è un linguaggio privo di tipo,i tipi di dati specificati per ogniuno degli attributi vengono usati solamente per chiarezza implementativa.

I Frames:
L'utilizzo di frames HTML nelle applicazioni web permette a più pagine di essere attive e visibili all'utente in un certo momento.Oggi è inoltre possibile avere più istanze browser attive sulla stessa macchina utente.Utilizzando scripts HTML dinamici e componenti HTML,tali pagine possono interagire l'una con l'altra.Il potenziale dal lato client per interazioni complesse è significante e necessita di essere modellato.
Per modellare i frames vengono introdotte altri due stereotipi di classe:*frameset* e *target* ed uno stereotipo di associazione:*targeted link*.Una classe frameset rappresenta un oggetto contenitore e si mappa direttamente nel tag HTML *frameset*;essa contiene le pagine client ed i targets.
La classe target è un frame con un nome o un'istanza browser referenziata da altre pagine client.
Un'associazione targeted link è un link ad un'altra pagina,ma quella che viene visualizzata in un target specifico.
Nell'esempio in figura 3.40 ,viene presentata una vista comune outline di un browser che utilizza due frames.Un frame viene chiamto con un target(Content),mentre l'altro frame contiene semplicemente una pagina client.Il frame della pagina client è la tabella dei contenuti del libro(TOC).I links di tale pagina sono targeted.Il risultato è una tabella statica dei contenuti nella parte sinistra della figura e i contenuti dei libri ,capitolo per capitolo,nella parte destra.
Molte delle attuali specifiche di presentazione vengono modellate attraverso valori etichettati nel frameset e nelle associazioni.Due valori etichettati nella relazione di aggregazione tra un franeset ed un target,o una pagina client,specifica a quale riga e quale colonna del frameset appartiene il target,o la pagina.
Quando un target non è aggregato con un frameset significa che viene usata un'istanza browser separate per visualizzare le pagine.

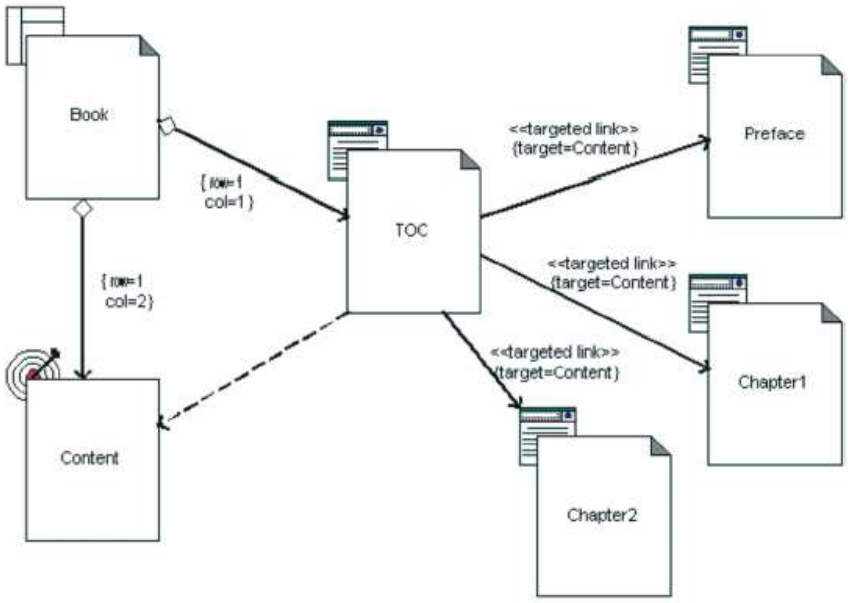

Figure 3.40: Esempio di modellazione di frames

La notazione di Conallen è stata adottata più o meno integralmente da diversi costruttori di ambienti CASE.
Ad esempio, *Rational Rose XDE* contiene alcuni profili UML specificamente concepiti per descrivere un'applicazione web.
Dai diagrammi di progetto,Rational Rose XDE è in grado di derivare template di pagina parziali ,che lo sviluppatore può completare a mano inserendo il codice che rappresenta la logica applicativa non modellabile in UML.
L'adozione di UML come notazione di specifica di applicazioni web costituisce un passo avanti nella maturazione del settore,piochè UML è uno standard diffuso e supportato da strumenti CASE di ottima qualità.
Tuttavia,UML di per sè non è sufficiente,in quanto l'estensione proposta da Conallen permette di utilizzare dei simboli costruttori di componenti ma le funzionalità di tali componenti devono poi essere programmate.Ciò rappresenta uno svantaggio dell'utilizzo di UML per la modellazione di interfacce web rispetto

a linguaggi di modellazione concepiti "ad hoc" per il Web e supportati da CASE tools specifici che permettono,nella maggiorparte dei casi,di generare le pagine web in modo automatico senza dover produrre alcun codice.

Analisi dell'estensione dell'UML di Conallen sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali dell'estensione dell'UML porposta da Conallen descritti precedentemente permettono di verificare se tale estensione soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,a differenza delle proposte descritte fin ora ,l'estensione dell'UML di Conallen non distingue tra livello di contenuto,di ipertesto e di presentazione,ma modella le pagine web dal lato client e dal lato server utilizzando classi UML stereotipate,come descritto precedentemente.
  L'ipertesto viene rappresentato attraverso i diagrammi della classi UML stereotipati e perciò la specifica dell'ipertesto tiene conto della distribuzione delle varie funzioni tra i diversi livelli dell'architettura web;a differenza ,ad esempio, di WebML o di altri linguaggi in cui il modello dell'ipertesto è più astratto e non considera gli aspetti architetturali.Tuttavia,proprio grazie ad un minore livello di astrazione,l'UML esteso di Conallen permette di modellare meccanismi tecnologici come gli scripts,le forms o i frames che fanno parte delle interfacce web e che i linguaggi descritti fin ora modellano solo in parte.

- Dal punto di vista dei *aspetti*,Conallen non discute alcuna modellazione del comportamento a parte le operazioni che possono essere definite insieme alle classi stereotipate.

- Dal punto di vista delle *fasi*,Conallen non propone nessuna fase di modellazione.

3.3.4 Koch-UWE

L'*UML-based Web Engineering (UWE)* [21] [23] descrive una metodologia sistematica per lo sviluppo di applicazioni Web basata su OOHDM ma che utilizza esclusivamente le tecniche UML,la notazione UML ed i meccanismi di estensione UML.
Allo scopo di rappresentare gli aspetti speciali del design di applicazioni web,UWE descrive come costruire modelli di navigazione e di presentazione definendo speciali elementi di modellazione UML stereotipati e valori etichettati,estendendo UML in modo analogo all'approccio di Conallen.
Inoltre UWE permette di definire altre viste utilizzando la varietà dei diagrammi UML come la modellazione dei tasks,la modellazione di scenari Web e la rappresentazione grafica della distribuzione dei componenti web.
Le attività principali della metodologia UWE sono dunque:l'analisi dei requisiti,la modellazione concettuale,la modellazione navigazionale e la modellazione della presentazione con in più la modellazione dei tasks e della distribuzione dei componenti delle applicazioni web e la visualizzazione degli scenari Web,le quali modellano gli aspetti dinamici dell'applicazione.
È inoltre prevista la generazione semi-automatica delle applicazioni web a partire da tali modelli.A tale scopo è stata implementata un'estensione del tool ArgoUML per supportare la costruzione dei modelli UWE;tali modelli verranno utilizzati per la generazione semi-automatica mediante un XML publishing framework [28].
La figura 3.41 mostra un diagramma delle classi che rappresenta il generico processo UWE,chiamato UWEXML.
Gli artefatti all'interno del processo di sviluppo vengono rap-

presentati come packages UML.Le dipendenze *trace* descrivono quali artefatti sono alla base di altri artefatti.
Il primo passo consiste nell'analisi e nel design dei modelli mediante il CASE tool UML esteso,ArgoUML.In segito tali modelli vengono trasformati dal Preprocessore UWEXML in rappresentazioni UML che vengono passate,insieme a documenti XML contenenti parametri per il processo di modellazione,al Generatore UWEXML.
Il generatore genera artefatti che possono essere direttamente distributi su componenti fisici,denotati dalla dipendenza *import*;ma anche artefatti che devono essere adattati prima della distribuzione su componenti fisici,denotati dalla dipendenza *refine*.
In tale processo si considera la distribuzione su un Application Server che fornisce un modello fisico dei componenti e su un XML publishing framework.
La metodolgia UWE risolve molti problemi della modellazione web utilizzando una notazione UML "pura",cioè priva di qualunque tipo di estensione.Gli aspetti speciali di un dominio ,come il web,vengono invece modellati utilizzando meccanismi di estensione che UML stesso fornisce,cioè utilizzando il così detto *profilo UML*.
La notazione utilizzata in UWE è un profilo UML "leggero" [22],cioè un profilo che può essere facilmente supportato dai tools.
I meccanismi di estensione UML utilizzati sono quelli utilizzati nell'estensione dell'UML di Conallen (sezione 3.3.3),ossia:

- *Stereotipi*:uno stereotipo UML è un nuovo tipo di elemento del modello definito all'interno di un modello basato su un tipo diverso di elementi esistenti.Gli stereotipi estendono i significati semantici ed hanno vincoli aggiuntivi ,ma non forniscono l'accesso al metamodello del linguaggio.L'intento

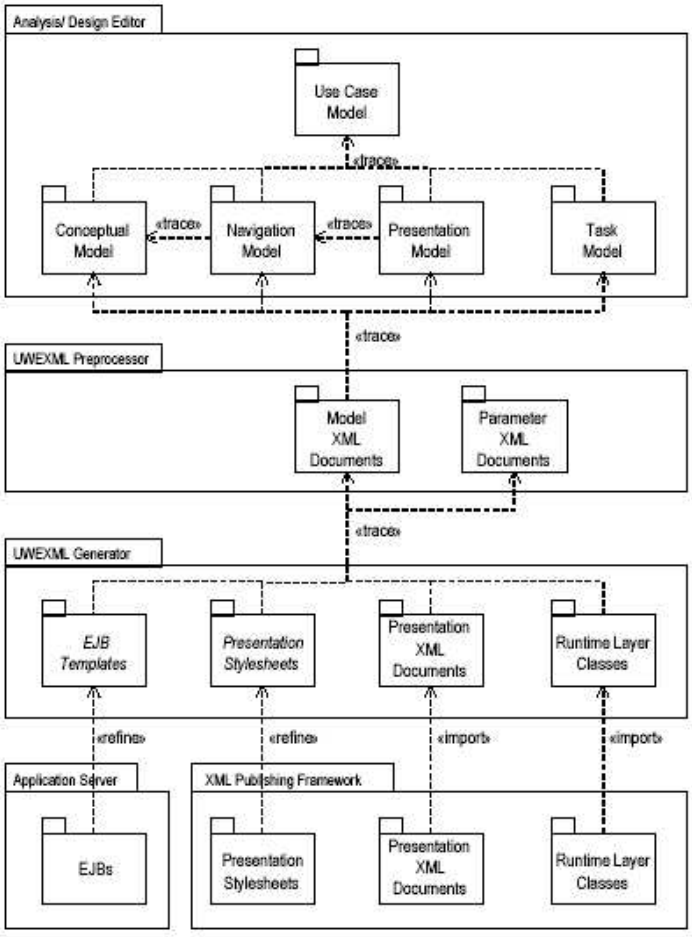

Figure 3.41: Il processo UWEXML

è che un generico tool di modellazione possa trattare un elemento stereotipato come un elemento di modellazione ordinario.L'uso del potente meccanismo degli stereotipi presenta sia vantaggi che rischi:
Il vantaggio è quello di creare con facilità linguaggi di modellazione per specifici domini applicativi con elementi di modellazioni definiti in modo più preciso ed espressivo.Il rischio è che l'eccessivo uso di stereotipi renda un linguaggio troppo difficile da utilizzare e da comprendere.

Alcuni stereotipi cambiano solamente la notazione di un elemento del modello e dunque servono come un tipo di commento;altri invece aggiungono o modificano le restrizioni semantiche del metamodello.

- *Valori etichettati*:Un valore etichettato UML è una coppia (etichetta,valore) che permette di associare un'informazione arbitraria ad ogni elemento del modello.
  L'informazione viene espressa nella forma di un testo ed è comunemente usata per esprimere requisiti non-funzionali o informazioni sulla gestione del progetto.
  UWE utilizza i valori etichettati per modellare la navigazione adattativa di applicazioni web personalizzate.La navigazione adattativa consiste in cambiamenti nella struttura di navigazione o nel modo di presentare tale struttura all'utente.

- *Vincoli*:un vincolo UML è una condizione o una restrizione che permette di specificare linguisticamente nuovi significati semantici per un elemento del modello.Tale specifica è data da un'espressione scritta in un linguaggio formale,come l'Object Constraint Language(OCL), oppure nel linguaggio naturale.

Specifica dei requisiti attraverso gli Use Cases UML

La metodologia UWE mostra come sia possibile specificare i requisiti per un'applicazione web attraverso lo strumento standard del diagramma Use Cases UML.
Un esempio per illustrare le tecniche UML utilizzate in UWE può essere il sito web di una libreria online personalizzata.
Tale applicazione di una libreria on line offre riguardo a pubblicazioni che possono essere giornali,libri oppure inproceedings.Ogni pubblicazione è descritta da un titolo,un numero,un editore,una

data di pubblicazione e da un insieme di articoli con i propri autori.Un libro viene considerato come un unico articolo il cui titolo è lo stesso del libro.Inoltre un ad ogni articolo e pubblicazione viene associato un insieme di parole chiave.Vengono distinti tre possibli tipi di utenti della libreria:i lettori registrati ,i lettori non.registrati e l'amministratore della libreria.IL lettore registrato viene modellato registrando gli articoli che visita ed egli può marcare gli articoli di suo interesse.L'applicazione mantiene una lista delle parole chiave personali di ogni lettore;esse possono essere sia positive che negative.
La figura 3.42 mostra una parte del diagramma Use Cases dell'applicazione libreria online,il quale può essere maggiormente dettagliato con un diagramma delle attività UML per specificare la sequenza delle azioni.

Modellazione concettuale attraverso il diagramma delle classi UML

Nella metodologia UWE viene utilizzato un diagramma delle classi UML per rappresentare graficamente il modello concettuale,cioè la visione statica degli elementi del dominio applicativo.
Analogamente ad OOHDM [17],UWE si propone di costruire il modello concettuale di un'applicazione web più indipendentemente possibile dagli aspetti di navigazione e presentazione.La figura 3.43 mostra il modello concettuale dell'esempio della libreria online limitatamente ai dati e alle funzionalità principali.

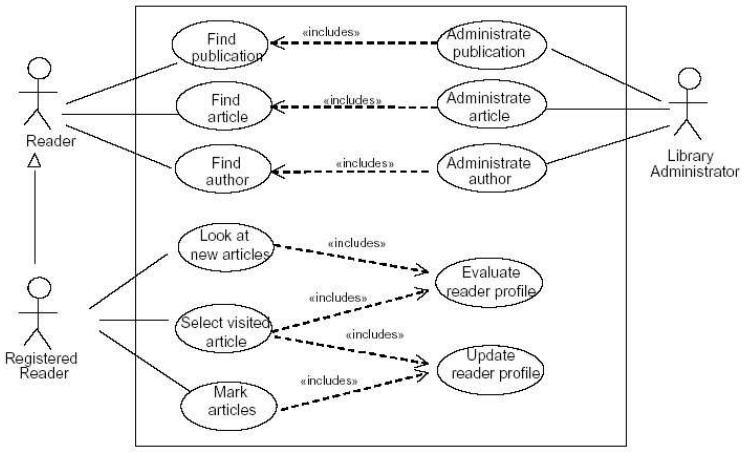

Figure 3.42: Modello Use Case di un'applicazione di libreria online

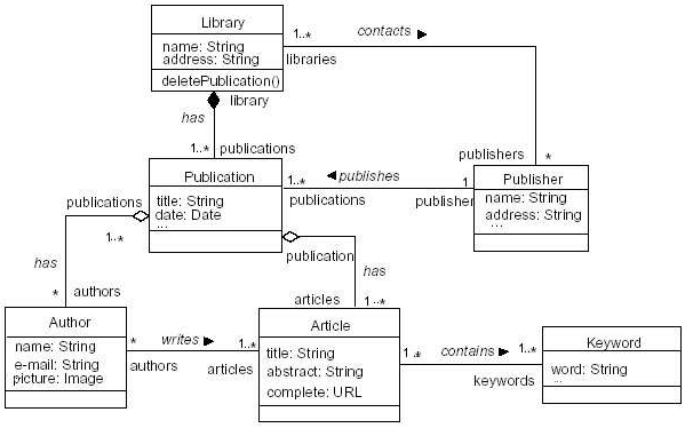

Figure 3.43: Modello concettuale di un'applicazione di libreria online

Modellazione della navigazione attraverso diagrammi delle classi stereotipati

La modellazione della navigazione di applicazioni web comprende la costruzione di due modelli di navigazione:

- Modello dello spazio di navigazione:specifica quali oggetti possono essere visitati mediante la navigazione;
- Modello della struttura di navigazione:specifica come tali

oggetti vengono raggiunti.

Il modello dello spazio di navigazione include le classi degli oggetti che possono essere visitati mediante la navigazione e le associazioni che specificano quali oggetti possono essere raggiunti.
La figura 3.44 mostra il modello dello spazio di navigazione per la libreria online.Gli elementi principali di tale modello sono la classe stereotipata *navigation class* e l'associazione stereotipata *direct navigability*.In tale modello sono incluse solo le classi del modello concettuale che sono considerate rilevanti per la navigazione,anche se l'informazione contenuta nelle classi omesse può essere mantenuta come attributi delle classi presenti(ad esempio,l'attributo publisher nella classe Pubblication).Vengono utilizzati vincoli in OCL per esprimere la relazione tra le classi concettuali e quelle di navigazione.

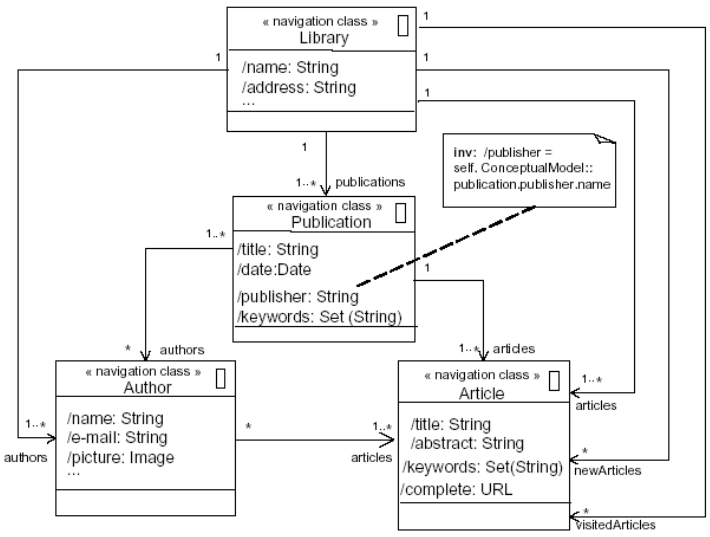

Figure 3.44: Modello dello spazio di navigazione di un'applicazione di libreria online

Il modello della struttura di navigazione viene costruito sulla base del modello dello spazio di navigazione;esso può essere con-

siderato come una fase di raffinamento .UWE mette a disposizione linee guida e meccanismi semi-automatici per tale raffinamento [23].
Le classi stereotipate per gli elementi di accesso sono *index*,*guided tour*,*query* e *menu* [22],le cui icone sono rappresentate in figura 3.45.

Figure 3.45: Icone degli stereotipi per gli elementi d'accesso

La figura 3.46 mostra il diagramma delle classi stereotipato che rappresenta il modello della struttura della navigazione dell'esempio della libreria online.Dalla figura si può osservare il valore etichettato *sorted* il quale indica che gli elementi dell'indice devono essere ordinati in base alle prefernze dell'utente ,che il sistema conosce mediante i valori correnti del modello utente.

Modellazione della presentazione attraverso diagrammi delle classi stereotipati

Per il modello di presentazione UWE utilizza una forma particolare di diagramma delle classi:si tratta di un diagramma delle classi che utilizza la notazione UML di composizione per le classi,cioè contenitori rappresentati graficamente da simboli delle parti all'interno del simbolo della classe.
Il modello di presentazione descrive dove e come gli oggetti della navigazione e le primitive d'accesso verranno presentate all'utente.Il modello di struutra della navigazione viene trasformato in un insieme di modelli che mostrano la locazione statica degli oggetti visibili dall'utente,cioè la rappresentazione schematica di tali oggetti (schizzi delle pagine).

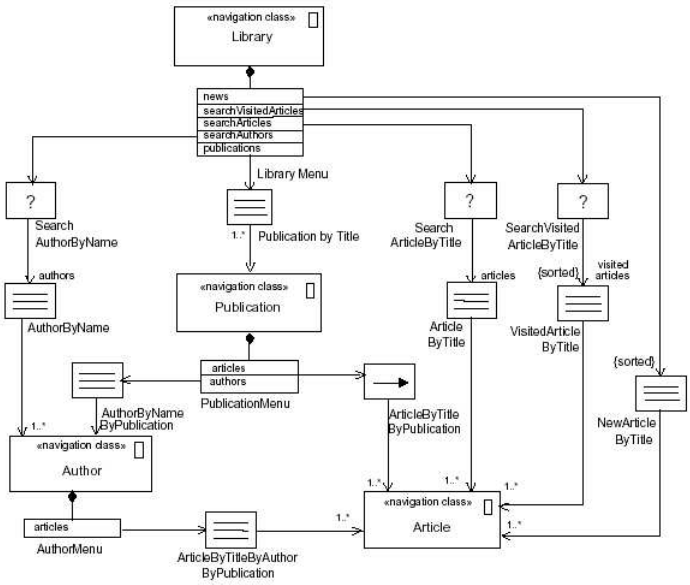

Figure 3.46: Modello della stuttura di navigazione di un'applicazione di libreria online

La figura 3.47 mostra lo schizzo di presentazione per la classe di navigazione pubblicazione.

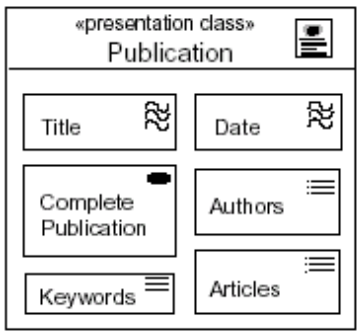

Figure 3.47: Schizzo della classe navigazionale Pubblicazione di una libreria online

L'insieme di elementi di modellazione stereotipati proposti in UWE per descrivere il modello di presentazione sono: *text*, *form*, *button*, *image*, *a*

collection ed *anchored collection.*Le ancore e le forms rappresentano gli elementi interattivi di base.

Modellazione degli scenari web attraverso statechart UML e diagrammi di interazione UML

La metodologia UWE si concentra sugli aspetti dinamici di un'applicazione Web,perciò individua quali dei diagrammi UML risultano più adatti per modellare tali aspetti.Uno di tali aspetti dinamici da modellare sono gli scenari di navigazione che UWE visualizza utilizzando i diagrammi UMl statechart(sequence diagrams), nello stesso modo del metodo OO-H [18],ma anche i diagrammi di interazione(collaboration diagrams).Ciò permette di dettagliare parti del modello della struttura di navigazione specificando gli eventi che causano le transazioni,definendo esplicitamente le condizioni e le azioni da svolgere.
La figura 3.48 mostra il diagramma statechart per l'interfaccia utente della libreria online,il fatto che vi sia un solo scenario permette di introdurre maggiori dettagli ,come determinare esplicitamente quando la navigazione nel verso opposto è possibile.
Un diagramma di interazione mostra l'interazione organizzata sulla base dei ruoli;esso è equivalente ad un diagramma statechart ma mostra le relazioni tra i ruoli,
Quale tipo di diagramma scegliere per la rappresentazione degli scenari web dipende dal livello desiderato di granularità.

Modellazione dei compiti attraverso i diagrammi delle attività UML

Un *compito (task)* è composto da uno o più sottocompiti o *azioni* che l'utente deve compiere per raggiungere un determinato obiettivo.Un obiettivo rappresenta un cambiamento desiderato nello stato del sistema e può essere realizzato componendo vari compiti e portandoli a termine.In UWE il concetto di *compito (task)* viene utilizzato in un senso pi ù ampio considerando compiti re-

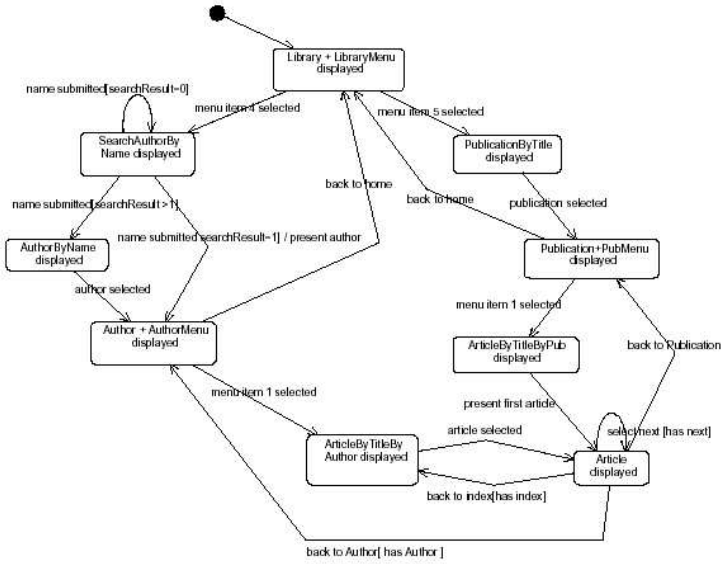

Figure 3.48: Diagramma statechart della presentazione web

alizzati dall'utente e compiti realizzati dal sistema.
Gli Use Cases dell'utente possono essere considerati come compiti,dunque ,poichè i diagrammi delle attività vengono utilizzati per dettagliare gli Use Cases,UWE utilizza i diagrammi delle attivit à per modellare i compiti(task).
Gli elementi di modellazione usati per modellare i compiti sono quelli dei diagrammi delle attività,cioè *attività*,*transizioni*,*diramazioni(branche* e *giunzioni(merges)*.
All'interno del modello di un compito viene utilizzata la dipendenza stereotipata *refine* tra le attività per indicare un livello di astrazione più fine.Inoltre In UWE viene scelta una distribuzione verticale dalle attività più grossolane a quelle di grana più fine allo scopo di rappresentare la gerrchia dei compiti.La modellazione dei compiti comprende anche la descrizione degli oggetti ,chiamati *referenti*,che l'utente dovrà percepire;tali oggetti sono gli oggetti concettuali e di presentazione inclusi nel modello dei compiti.La relazione tra tali oggetti ed i compiti viene espressa

come un oggetto flusso visualizzato con una linea tratteggiata direzionata.
Vengono usati oggetti di presentazione "ingoing" per esprimere l'input dell'utente attraverso tali oggetti,mentre oggetti di presentazione "outgoing" sono usati per esprimere l'output verso l'utente.Inoltre,gli oggetti concettuali vengono utilizzati per esprimere l'input e l'output dei compiti.
La figura 3.49 mostra il modello di compito per il compito"Cancella pubblicazione".

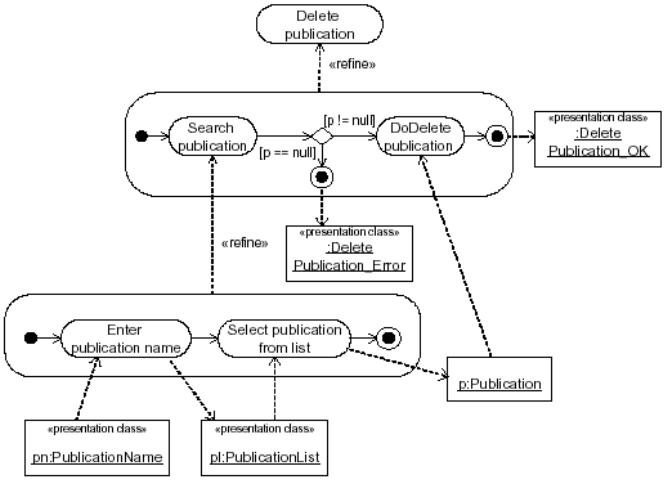

Figure 3.49: Modello del compito "Cancella pubblicazione" mediante diagramma delle attività

Rappresentazione della distribuzione dei componenti web attraverso il diagramma di deployment UML

UWE utilizza i diagrammi di deployment UML per documentare la distribuzione dei componenti di un'applicazione Web.Gli elementi principali dei diagrammi di deployment UML sono i *nodi*,rappresentati graficamente da cubi.Un nodo è un elemento fisico che esiste a run-time e rappresenta una risorsa computazionale.Un nodo può contenere oggetti e *componenti* che si trovano all'interno della

risorsa computazionale.
Un componente UML è una parte fisica del sistema che permette la realizzazione di un insieme di interfacce ed è rappresentato con un rettangolo.Inoltre è possibile modellare anche connessioni fisiche tra i nodi come,per esempio,le connessioni TCP/IP. La figura 3.50 mostra una parte de diagramma di deployment dell'esempio della libreria online.In particolare vengono considerati l'elemento Pubblicazione e le relazioni di dipendenza tra i componenti.La parte superiore della figura mostra come i componenti dell'applicazione di esempio possono essere distribuiti;nella parte inferiore vengono aggiunte classi del modello di design dell'applicazione per mostrare mediante quali componenti fisici verranno realizzate le classi.
L'elemento Pubblicazione del modello concettuale verrà realizzato da un componente Enterprise Java Beans(EJB) che si trova nel nodo dell'Application Server.L'elemento Pubblicazione dal modello di presentazione viene realizzato da un componente Java Server Page(JSP) che si trova nel nodo del Web Server;esso utilizza il componente EJB e ciò viene modellato mediante la dipendenza *use*.Infine all'interno del nodo del Web Browser viene mostrato all'utente il componente pagina client[19] per l'elemento Pubblicazione.

Analisi della metodologia UWE sulla base dei requisiti legati alle tre dimensioni ortogonali

Gli elementi fondamentali della metodologia UWE descritti precedentemente permettono di verificare se tale metodologia soddisfa o meno i requisiti dei linguaggi di modellazione web categorizzati sulla base di tre dimensioni ortogonali nel capitolo 2,sezione 2.2.

- Dal punto di vista dei *livelli*,la metodolgia UWE,pur basandosi sullo stesso tipo di esntesione di UML di Conallen,mantiene

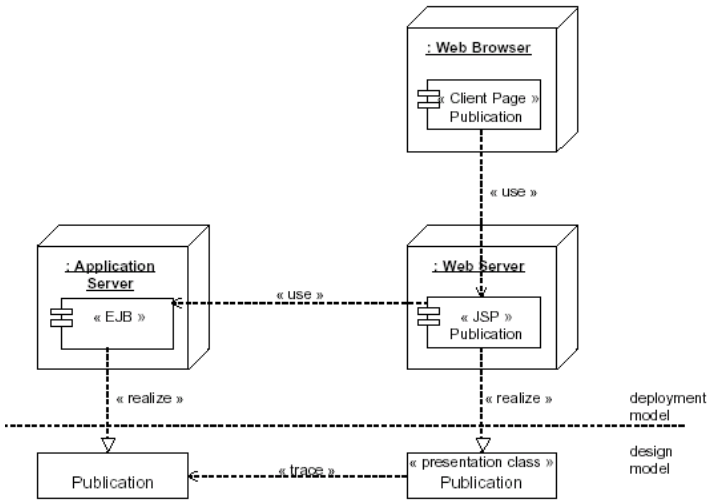

Figure 3.50: Diagramma di deployment basato sull'elemento Pubblicazione dell'esempio di libreria online

la distinzione tra i tre livelli di contenuto,ipertesto e presentazione che in parte coincidono con le fasi della metodologia.
A livello di contenuto,viene utilizzato un diagramma delle classi UML,chiamato *modello concettuale*,senza nessuna estensione.
Il livello di ipertesto viene modellato per mezzo di due diagrammi descritti precedentemente:il*modello dello spazio di navigazione*ed il *modello della struttura di navigazione*.
Si tratta di diagrammi delle classi UML estesi per supportare la modellazione di applicazioni web mediante meccanismi come stereotipi,valori etichettati e vincoli.
A livello di presentazione UWE utilizza un particolare tipo di diagramma delle classi,il *modello di presentazione statico*,per modellare come gli oggetti di navigazione verranno presentati all'utente ed i diagrammi statechart di UML,il *modello di presentazione statico*, per descrivere l'attivazione della navigazione e la trasformazione dell'interfaccia utente in

base alle reazioni agli eventi.
L'approccio UWE non discute il mapping tra il livello di ipertesto e quello di presentazione.

- Dal punto di vista dei *aspetti*,il comportamento dinamico delle applicazioni web viene modellato mediante i diagrammi UML di interazione,di collaborazione ,di sequenza e statechart per rappresentare gli scenari Web ed i tasks. Tali diagrammi possono essere collocati a livello di presentazione,dunque l'aspetto comportamentale viene trattato solo a tale livello. L'aspetto strutturale viene trattato ad ogni livello.

- Dal punto di vista delle *fasi*,la metodologia UWE fa riferimento ad un preciso processo UWEXML e all'interno del primo passo,quello di analisi e design, individua varie fasi di creazione di modelli che coincidono in parte con la distinzione tra i livelli :dalla modellazione dei requisiti ,dei contenuti,della navigazione e della presentazione,alla rappresentazione della distribuzione dei componenti di un'applicazione web.

3.4 Conclusioni dell'analisi dei linguaggi di modellazione web sulla base dei requisiti legati alle tre dimensioni ortogonali

Dalle valutazioni di ogni linguaggio effettuate facendo riferimento ai requisiti categorizzati sulla base delle tre dimensioni ortogonali (livelli,aspetti e fasi;capitolo 2,sezione 2.2) sono emerse le seguenti limitazioni per i linguaggi do modellazione analizzati:

- *Mancanza di mapping esplicito e flessibile*:presente nei linguaggi delle prime generazioni nei quali ,benchè sia presente

una distinzione tra i livelli di contenuto,ipertesto e presentazione,non vengono descritti concetti per il mapping tra i livelli.
Nelle proposte di ultima generazione tale mancanza viene in parte colmata per quanto riguarda il mapping tra contenuto ed ipertesto (Araneus,WebML),anche se rimane in modo particolare per il mapping tra ipertesto e presentazione.

- *Mancanza della modellazione dell'aspetto comportamentale*:presente in particolare nei linguaggi meno recenti i quali considerano tale aspetto solo a livello di presentazione.
  Tra le proposte più recenti,quelle basate su UML (come UWE,Conallen,
  Metodo OO-H),insieme con l'approccio di WebML, considerano maggiormente l'aspetto comportamentale anche a livello di ipertesto.

- *Assenza di un formalismo comune di modellazione*:tutti i linguaggi analizzati sono basati su di un'insieme dei principali formalismi di modellazione,fatta eccezione per la metodologia UWE e l'estensione dell'UML di Conallen i quali si basano completamente su UML.

- *Supporto dei patterns solo a livello di ipertesto*:non vengono forniti concetti per supportare la modellazione dei patterns a tutti i livelli,fatta eccezione per OOHDM ed in particolare per la sua evoluzione,il metodo OO-H ,il quale mantiene un catalogo dei patterns utilizzati come supporto per la modellazione sia a livello di ipertesto che di presentazione.

- *Mancanza di modellazione concettuale e logica a livello di presentazione*:
  molti dei metodi di modellazione non supportano il livello di presentazione con adeguati concetti di modellazione concettuale e logica.

Piuttosto vengono spesso proposti specifici tools per la gestione del livello di presentazione,perdendo così il beneficio dell'indipendenza dalla tecnologia.

- *Mancanza di un processo di sviluppo*:molti dei primi linguaggi proposti non seguono un processo che guidi le varie attività dello sviluppo di applicazioni ipermediali;mentre tutte le proposte più recenti fanno parte di progetti basati su processi di sviluppo completi.

3.5 Valutazione dei linguaggi di modellazione web sulla base dei requisiti individuati nel capitolo 2, sezione 2.1

I linguaggi per la modellazione di interfacce web descritti possono essere analizzati e confrontati facendo riferimento ai requisiti categorizzati sulla base dell'architettura informativa e funzionale (capitolo 2, sezione 2.1).
A tale scopo sono state costruite delle tabelle che mostrano se i linguaggi di modellazione web analizzati presentano o meno i requisiti auspicati (Legenda delle tabelle:S=sì,N=no,P=parzialmente).

3.5.1 Analisi sulla base dei requisiti funzionali

La valutazione sulla base dei requisiti funzionali (paragrafo 2.1.1) è documentata nella tabella 3.2,per la prima e seconda generazione di linguaggi, e nella tabella 3.3,per la terza generazione.

3.5.2 Analisi sulla base dei requisiti legati all'informazione

Le tabelle 3.4 e 3.5 documentano,rispettivamente, la valutazione dei linguaggi di modellazione web delle prime generazioni e di

Requisito	HDM	RMM	HDM-Lite	OOHDM
Modellazione di integrazione e connettività	N	N	P	N
Supporto modellazione di patterns	N	N	N	S
Rappresentazione indipendente dalla tecnologia	N	N	S	P

Table 3.2: Analisi dei linguaggi di prima e seconda generazione sulla base dei requisiti funzionali

Requisito	Strudel	Araneus	WebML	UWE	OO-H	Conallen
Modellazione di integrazione e connettività	N	N	N	P	P	P
Supporto modellazione di patterns	N	N	N	N	S	P
Rappresentazione indipendente dalla tecnologia	N	P	S	S	S	P

Table 3.3: Analisi dei linguaggi di terza generazione sulla base dei requisiti funzionali

quelli più recenti sulla base dei requisiti legati all'informazione (paragrafo 2.1.2).

Requisito	HDM	RMM	HDM-Lite	OOHDM
Modellazione concetti di presentazione	P	P	S	S
Modellazione della navigazione	P	S	S	S
Modellazione interazioni utente-informazione	N	P	S	P
Modellazione dei ruoli degli utenti	N	N	S	N
Modellazione del contenuto	P	S	S	S

Table 3.4: Analisi dei linguaggi di prima e seconda generazione sulla base dei requisiti legati all'informazione

3.5.3 Analisi sulla base dei requisiti di carattere generale

Le tabelle 3.6 e 3.7 documentano la valutazione dei linguaggi di modellazione web sulla base dei requisiti di carattere generale (paragrafo 2.1.3).

Requisito	Strudel	Araneus	WebML	UWE	OO-H	Conallen
Modellazione concetti di presentazione	P	P	S	S	S	N
Modellazione della navigazione	S	S	S	S	S	P
Modellazione interazioni utente-informazione	S	P	S	S	S	P
Modellazione dei ruoli degli utenti	N	N	S	N	S	P
Modellazione del contenuto	S	S	S	S	S	P

Table 3.5: Analisi dei linguaggi di terza generazione sulla base dei requisiti legati all'informazione

3.5.4 Le limitazioni dei linguaggi analizzati

Sulla base dei risultati ,documentati dalle tabelle, per quanto riguarda il confronto tra i linguaggi di modellazione web analizzati ed i requisiti auspicabili,si possono individuare alcune limitazioni principali degli approcci esistenti per la modellazione di interfacce web:

- *Disconnessione tra architettura funzionale ed architettura informativa*:
  Con l'evoluzione delle applicazioni web dai siti tradizionali "read-only" contenenti informazione ad applicazioni distribuite,risulta sempre più importante collegare capacità di navigazione e di visualizzazione con operazioni e transazioni.Infatti le applicazioni web possono essere viste come:

  - Un'estensione dei sistemi informativi tradizionali ,arricchiti con la navigazione e con strutture informative

Requisito	HDM	RMM	HDM-Lite	OOHDM
Collegare funzionalità ed informazione	N	N	N	N
Mantenere l' integrità del sistema	N	N	P	N
Modellazione dei vari livelli di astrazione	P	P	S	S
Supporto della gestione del ciclo di vita	N	N	P	P
Supporto di un CASE tool	N	S	S	N

Table 3.6: Analisi dei linguaggi di prima e seconda generazione sulla base dei requisiti di carattere generale

complesse.

- Un'estensione dei siti web tradizionali arrichiti con vari tipi di operazioni e funzionalità convenzionali dei sistemi informativi.

La modellazione degli elementi ipermediali tradizionali(navigazione e presentazione dei contenuti) e la modellazione dell'aspetto dinamico delle operazioni non devono essere condotte separatamente ,ma si devono integrare ed estendere le metodologie e le tecniche di modellazione provenienti dall'ambito dell'ipermediale e dall'ambito della modellazione "tradizionale" delle operazioni.

Benchè l'integrazione tra le strutture informative ipermediali e gli elementi operazionali risulti di fondamentale importanza per le applicazioni web,tale integrazione non viene

Requisito	Strudel	Araneus	WebML	UWE	OO-H	Conallen
Collegare funzionalità ed informazione	N	N	N	N	N	N
Mantenere l' integrità del sistema	P	P	P	N	P	N
Modellazione dei vari livelli di astrazione	S	S	S	P	S	P
Supporto della gestione del ciclo di vita	P	P	P	P	P	P
Supporto di un CASE tool	P	S	S	P	S	P

Table 3.7: Analisi dei linguaggi di terza generazione sulla base dei requisiti di carattere generale

realizzata in nessuno dei linguaggi analizzati.
Per quanto riguarda i linguaggi che hanno origine nell'ambito ipermediale,come HDM, OOHDM, HDM-Lite e WebML,essi usualmente forniscono elementi sofisticati per supportare gli aspetti informativi,in particolare la modellazione della navigazione e della presentazione del contenuto informativo.
Tuttavia tali linguaggi offrono capacità di modellazione molto limitate per quanto riguarda la modellazione funzionale.
In contrasto,l'approccio di Conallen si propone di estendere la notazione UML allo scopo di supportare la modellazione delle operazioni e delle interazioni all'interno delle applicazioni web,ma l'aspetto informativo non viene modellato in modo adeguato.
Infatti Conallen modella,pur con la limitazione della notazione UML di base, le operazioni ed i meccanismi tecno-

logici che riguardano le applicazioni web,ma non fornisce elementi adeguati per la modellazione di navigazione e presentazione.

- *Cattivo uso dei meccanismi di estensione UML*:gli approcci che adattano le tecniche di modellazione object-oriented,ed in particolare la notazione UML,alla modellazione di applicazioni web(approcci di Conallen e Koch-UWE) mostrano una maggiore capacità di modellare l'aspetto funzionale.
  Tuttavia essi risultano fortemente limitati dal fatto che UML rappresenta una modellazione ad oggetti "generale" che non è in grado di cogliere le peculiarità della modellazione web,nemmeno nelle sue estensioni proposte basate sull'uso di stereotipi.

- *Parziale supporto di CASE tools*:alcuni dei linguaggi analizzati presentano un supporto parziale degli strumenti di CASE in quanto non permettono la generazione automatica del codice necessario per le applicazioni web ma richiedono un'attività di scrittura del codice.
  Si tratta in particolare delle proposte di estensione UML di Conallen e Koch,le quali fanno riferimento a Rational Rose XDE o ad un XML publishing framework ,e del progetto Strudel, nel quale i templates HTML delle pagine devono essere scritti dal programmatore.

- *Incapacità di supportare la gestione dell'intero ciclo di vita*:molti dei linguaggi di modellazione web di ultima generazione analizzati fanno riferimento a processi che si concentrano sul fornire notazioni per le fasi di modellazione ed implementazione,mentre altre fasi del ciclo di vita delle appli-

cazioni web come la specifica dei requisiti ed il mantenimento non vengono supportate in modo appropriato.

3.5.5 Conclusioni

Come emerso dai limiti evidenziati,lo stato dell'arte dei linguaggi di modellazione Web non fornisce una notazione chiara ed uniforme in grado di rappresentare ad un alto livello di astrazione sia il contenuto informativo che le funzionalità delle moderne applicazioni web.
In particolare,la maggior parte dei linguaggi proposti,di origine ipermediale,si concentra sulla modellazione della navigazione e della presentazione dell'informazione in termini di pagine e links dell'ipertesto,ma non permette di rappresentare le funzionalità dei componenti front-end che realizzano i servizi che le applicazioni web devono fornire e la loro interazione ed integrazione.
Al contrario,le proposte di estensione dell'UML,in particolare l'UML esteso di Conallen, permettono di modellare in parte tali funzionalità,ma non forniscono strumenti adeguati per la visualizzazione e la navigazione delle pagine contenenti linformazione.
Inoltre dall'analisi condotta emerge la mancanza di un approccio che sia in grado di modellare gli aspetti informativi,ma sopratutto gli aspetti funzionali,di un'applicazione web ad un alto livello di astrazione senza essere legato a dettagli implementativi.Infatti linguaggi come HDM-lite,WebML, Metodo OO-H ,pur limitati nella rappresentazione delle funzionalità,ragionano ad un alto livello di astrazione;mentre l'approccio di Conallen ,che coglie maggiormente gli aspetti funzionali,risulta più vincolato a dettagli architetturali ed implementativi.
Per quanto riguarda il supporto di CASE tools,tutti gli approcci di ultima generazione sono stati adottati più o meno integralmente da costruttori di ambienti CASE o includono strumenti

di CASE specifici.
Da questo punto di vista,si può pensare di valutare in modo positivo i linguaggi supportati da CASE tools che realizzano la generazione automatica del codice,riducendo drasticamente i tempi di implementazione.
In conclusione non è possibile individuare il linguaggio di modellazione web "migliore",ma solamente quello più idoneo a seconda delle caratteristiche dell'applicazione web che si deve modellare.

Chapter 4

Strumenti di sviluppo web

Negli ultimi anni è stata proposta una miriade di prodotti e di soluzioni per lo sviluppo di applicazioni web;dalle tecnologie principali che permettono lo sviluppo di applicazioni nel web(come le architetture J2EE e .NET),ai rapidi strumenti di sviluppo di applicazioni e alle soluzioni verticali.

Limitando l'interesse agli strumeti di sviluppo web,possono essere individuate le seguenti categorie di prodotti:

- **Sistemi di gestione del contenuto:**questi prodotti considerano le applicazioni web principalmente come applicazioni per la pubblicazione e la gestione del contenuto informativo nell'ambito web e cosìoffrono funzionalità per la rapida costruzione di ipertesti per l'inserimento e la pubblicazione di contenuti.

- **Scheletri verticali:**questi prodotti inglobano applicazioni parzialmente istanziate(per esempio,portali web),le quali possono essere estese per permettere la soddisfazione di requisiti specifici di un'organizzazione.

- **Sistemi orizzontali component-based:**questi prodotti uniscono alcuni componenti e wizards utili i quali offrono l'infrastruttura di base per costruire unapplicazione web;tali

prodotti sono indipendenti dal dominio verticale e possono essere usati per sviluppare diverse categorie di applicazioni web

- **Strumenti RAD:**questi strumenti si basano sull'abilità dello sviluppatore nella fase di implementazione ed offrono funzioni integrate per implementare i vari livelli di un'applicazione web:presentazione,logica di business e dati.

- **Strumenti di sviluppo web model-driven e generatori di codice:**
questi prodotti estendono la funzione degli strumenti RAD anche alle ultime fasi del ciclo di vita dell'applicazione,incorporando l'analisi dei requisiti,la modellazione visuale e la generazione del codice.

In particolare gli strumenti di sviluppo web model-driven e generatori di codice permettono di applicare anche al web i vantaggi dell'approccio model-driven evidenziati già in altri ambiti del software.
Tale categoria viene considerata come la probabile evoluzione dell'attuale mercato degli strumenti di sviluppo web,caratterizzato da un gran numero di prodotti principalmente concentrati sul supporto dell'implementazione.
Gli strumenti di sviluppo model-driven vengono dunque trattati in modo approfondito in questo capitolo,presentando l'applicazione ad un caso pratico di due esempi significativi.

4.1 Strumenti di sviluppo model-driven

Il termine "strumenti di sviluppo model-driven" fa riferimento a quegli strumenti di sviluppo che comprendono funzioni per supportare non solo la fase di implementazione delle applicazioni web,ma anche la fase di analisi e design,utilizzando metodologie

e notazioni formali.
Gli strumenti appartenenti alla categoria "model-driven" possono essere suddivisi ,in modo approssimativo,in tre ampie aree:

- **Strumenti basati sui Dati:** derivano da prodotti antecedenti al web,
  usati per il design di sistemi informativi.Questi prodotti sono caratterizzati da un'estensione proprietaria delle notazioni di modellazione dei dati (per esempio,il modello E/R) per supportare la modellazione del Web.
  Come prodotto rappresentativo di questa classe si può citare *Oracle 9i Designer*,ma anche il *CASE tool HOMER* supporto della metodologia Araneus (paragrafo 3.2.2).

- **Strumenti Object-Oriented:** hanno origine dai prodotti che supportano metodologie per l'analisi ed il design OO,utilizzati per lo sviluppo di applicazioni object-oriented.
  I prodotti di questa classe sono caratterizzati dall'uso di UML,esteso per modellare i componenti delle applicazioni web.
  Come prodotti rappresentativi di questa classe si possono citare
  OracleJDeveloper e *Rational Rose XDE.*

- **Strumenti specifici del Web:** sono prodotti concepiti appositamente per l'analisi,il design e la generazione semi-automatica/automatica di applicazioni web.Essi adottano metodologie,tecniche e notazioni inventate proprio per questo obiettivo.
  Come prodotto rappresentativo di questa classe si può citare *WebRatio Site Development Studio.*

4.1.1 Oracle 9i Designer

Oracle Designer rappresenta un precursore dei sistemi CASE per il Web;l'idea originale di questo ambiente di sviluppo è quella di generare applicazioni Web mediante l'estensione dei diagrammi *Entità/Relazione (E/R)*,quindi con un approccio centrato sui databases.
Oracle Designer è un ambiente per il processo di business e la modellazione di applicazioni,integrato con generatori di codice originariamente progettati per ambienti client-server tradizionali.
La famiglia di tools di *Oracle Designer* comprende quattro categorie che riflettono le necessità di diversi tipi di utenti:

- L' *area di modellazione dei requisiti del sistema* include strumenti per modellare i requisiti ed i processi di business per mezzo di diagrammi E/R dei dati ,funzioni e flussi di dati nel sistema.

- L' *area di trasformazione di designs preliminari* include un insieme di Trasformatori che generano designs preliminari a partire dai modelli precedentemente creati.Per esempio,vi sono trasformatori per la creazione del database dagli schemi E/R e trasformatori per il design dell'applicazione.

- L' *area di design e di generazione* rappresenta il componente della suite piùinteressante per la modellazione Web in quanto include un editor di specifiche visuali ma sopratutto un generatore di codice,il *Generatore Oracle Web PL/SQL*,il quale è in grado di creare applicazioni Web dinamica mediante la formulazione di queries,la modifica e l'inseriemento di informazione in un database relazionale Oracle.

- L' *area degli strumenti di deposito* contiene il deposito delle

risorse definite e fornisce un vasto insieme di primitive di amministrazione e controllo.

Oracle Designer non sfrutta una specifica notazione per la modellazione ad alto livello dell'ipertesto,ma utilizza un insieme di notazioni per la modellazione di sistemi informativi.
Il limite dell'approccio di *Oracle Designer* è nel suo forte orientamento ai dati .Esso può essere usato per generare i componenti software per la pubblicazione Web a partire da un vasto insieme di modelli concettuali,ma il modello E/R su cui si basa,per quanto esteso,non si presta a descrivere gli aspetti che caratterizzano un'applicazione Web ,specialmente la navigazione e la presentazione delle pagine.

4.1.2 OracleJDeveloper

OracleJDeveloper è l'ultimo ambiente di sviluppo proposto da Oracle;esso supporta lo sviluppo ,il debugging e l'implementazione di applicazioni Web.
JDeveloper include un insieme di tools integrati per supportare il ciclo di vita completo delle applicazioni web,dalla modellazione alla generazione di codice.
JDeveloper è orientato verso uno sviluppo in ambiente J2EE ed include wizards e strumenti visuali di design per creare componenti J2EE come JavaServer Pages(JSP),servlets ed Enterprise JavaBeans(EJB).

JDeveloper include anche un *UML Class Modeler* il quale permette di modellare le classi Java mediante diagrammi delle classi UML , e di generare tali classi.Il *Modeler* supporta inoltre la sincronizzazione del modello e del codice ed il reverse-engineering del codice Java esistente in modelli.

Analogamente ad *Oracle 9i Designer*,anche *JDeveloper* non ha alla base una metodologia che permetta di rappresentare i componenti citati ad un alto livello di astrazione senza essere vincolati dall'ambiente di sviluppo di destinazione.

4.1.3 WebRatio Site Development Studio

WebRatio Site Development Studio è uno strumento di CASE sviluppato al Politecnico di Milano come supporto al linguaggio Web Modelling Language (WebML,vedi paragrafo 3.1.3) per la modellazione concettuale di applicazioni web.

WebRatio IDE

WebRatio fornisce un'ambiente integrato di sviluppo (Integrated Development Environment) per la modellazione di applicazioni web attraverso il modello E/R ed il linguaggio visuale WebML. Infatti l'interfaccia grafica di *WebRatio* permette di costruire in modo visuale i diagrammi che descrivono la struttura dell'informazione che l'applicazione deve gestire ed i diagrammi degli ipertesti delle viste di sito operanti su tale informazione.
Un aspetto innovativo di *WebRatio* è la cura particolare dedicata agli aspetti di presentazione,cioè alla disposizione dei contenuti nella pagina e alla veste grafica. *WebRatio* può importare,mediante il tool EasyStyler in esso integrato, esempi di grafica di alta qualità definiti in un linguaggio di markup compatibile con XML e trasformarli automaticamente in fogli di stile XSL ,cioè in micro-compilatori in grado di produrre templates di pagina conformi sia al modello concettuale dell'applicazione sia alla veste grafica prescelta.

L'interfaccia utente di *WebRatio Site Development Studio*,mostrata in figura 4.1,è strutturata nelle aree tipicamente messe a disposizione dai più popolari ambienti integrati di sviluppo(IDEs):

- La *barra del menu*:contiene il menu principale.
- La *barra degli strumenti*:contiene le icone dei comandi usati più frequentemente.

- L'*albero del progetto*:collega tutti gli elementi che costituiscono il progetto di un'applicazione,organizzati in modo gerarchico.
- L'*area di lavoro*:rappresenta l'area principale dell'interfaccia grafica dove possono essere rappresentate graficamente le specifiche visuali dell'applicazione.
- Il *frame delle proprietà*:mostra le proprietà dell'elemento selezionato nell'albero del progetto o nell'area di lavoro e permette di modificare tali proprietà.
- Il *frame degli errori/ricerca*:rappresenta un'area dedicata alla visualizzazione di messaggi ,come errori,warnings,o i risultati di una ricerca.

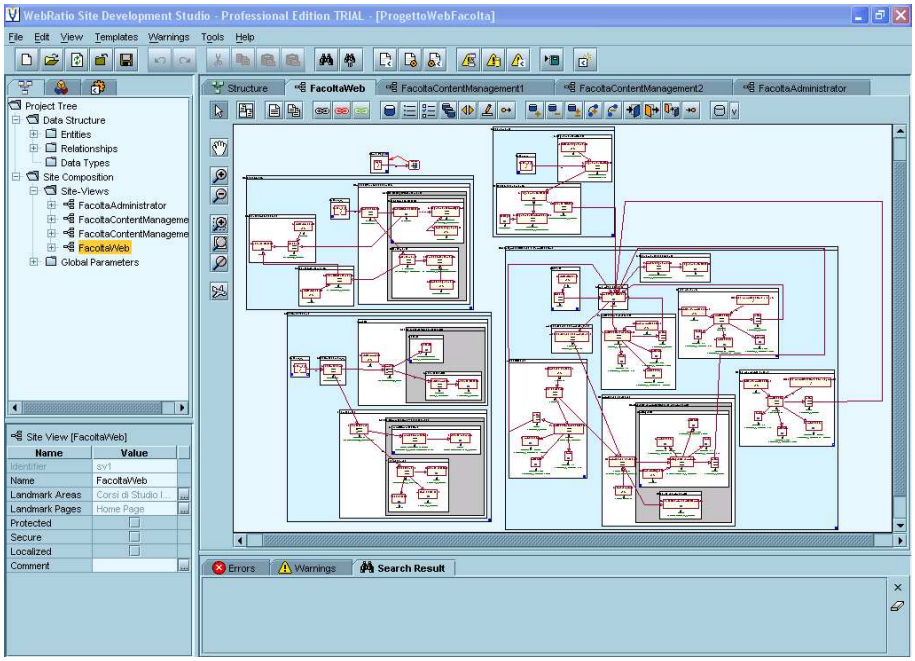

Figure 4.1: Le aree dell'interfaccia utente di WebRatio Site Development Studio

Il contenuto ed il funzionammento dell'albero di progetto e dell'area di lavoro dipendono dalla *vista di progetto* selezionata.

Le viste di progetto sono prospettive diverse del progetto WebRatio;è possibile selezionare una di tali viste per realizzare scopi differenti nel processo di sviluppo di un'applicazione :

- *Vista di Editing*:supporta il design dello schema strutturale e delle site views.
- *Vista di Mapping*:permette di definire e gestire le risorse del progetto ,connettendosi con un database esistente o generando un nuovo database di supporto ,possibilmente contenente dati di prova.
- *Vista di Presentazione*:permette di applicare i fogli di stile alle pagine , decidere la posizione delle content units all'interno della pagina e generare i template di pagina.

È possibile cambiare la vista di progetto attiva cliccando sulle icone sopra all'albero di progetto(figura 4.2).

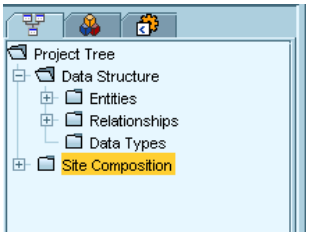

Figure 4.2: Le tre prospettive dell'albero di progetto

Come citato precedentemente, i comandi usati più frequentemente sono forniti come icone nella barra principale degli strumenti(figura 4.3).

Figure 4.3: La barra degli strumenti principale

Oltre alla barra principale degli strumenti,WebRatio offre barre degli strumenti specifiche,posizionate all'interno dell'area di lavoro.I comandi presenti in tali barre degli strumenti sono specifici del lavoro che si sta realizzando:

- *Barra degli strumenti per l'Editing della Struttura*:contiene i comandi per rappresentare il diagramma strutturale (figura 4.4).

Figure 4.4: La barra degli strumenti per l'editing della struttura

- *Barra degli strumenti per l'Editing delle Site Views*:contiene i comandi per rappresentare i diagrammi ipertestuali delle Site Views (figura 4.5).

Figure 4.5: La barra degli strumenti per l'editing delle site views

Generatore di codice WebRatio

Il cuore di *WebRatio* è un generatore di codice basato su XML che permette di trasformare automaticamente le specifiche WebML dell'applicazione,codificate in XML,nel codice finale dell'applicazione per le piattaforme J2EE e Microsoft.NET.
Infatti l'edizione iniziale di *WebRatio Site Development Studio* si basa sulla piattaforma J2EE,ma recentemente è stata sviluppata anche l'edizione .NET.
Il codice generato copre tutti gli aspetti di un'applicazione web dinamica:le queries SQL per l'estrazione delle informazioni dalle fonti dati,il codice per la logica di business dell'applicazione ,i templates di pagina per la costruzione dinamica delle interfacce utente ed i descrittori XML.

Tutte le applicazioni generate con *WebRatio*(edizione J2EE) condividono la stessa architettura software ,corrispondente alla classica architettura a tre livelli,nella quale l'architettura del livello di presentazione sfrutta un pattern consolidato nel design software,chiamato *MVC (Model View Controller)*.

L'architettura MVC si basa sulla separazione delle tre funzioni essenziali riguardanti la presentazione dell'informazione e l'interazione con l'utente:

- L'interazione con il livello di business per soddisfare le richieste del client (il Modello).
- Il controllo dell'interazione per mezzo delle azioni dell'utente (il Controllore).
- L'interfaccia presentata all'utente (la Vista).

Nell'architettura MVC l'utente effettua una richiesta per qualche contenuto o servizio,il Controllore intercetta la richiesta e decide quale azione deve essere compiuta per soddisfare tale richiesta.Il Controllore invia la *richiesta d'azione* al corrispondente elemento del Modello.Il Modello esegue tale azione interagendo con i servizi del livello di business;ciò comporta un cambiamento nello stato dell'applicazione e produce un risultato che deve essere comunicato all'utente.Il cambiamento nel Modello attiva la Vista più appropriata che crea la presentazione del risultato.

In *WebRatio* l'originale MVC viene adattata tenendo conto delle peculiarità del contesto Web.
La figura 4.6 mostra l'architettura comune a tutte le applicazioni generate con WebRatio(edizione J2EE).

Ogni pagina WebML viene mappata in quattro elementi:

1. Un'*azione di pagina* nel Modello.
2. Un *servizio di pagina* nel livello di business.
3. Un *template JSP* nella Vista.
4. Un *mapping dell'azione* nel file di configurazione del controllore.

L'*azione di pagina* è un'istanza di una classe Java,a volte chiamata*classe d'azione* ,invocata dal Controllore quando l'utente richiede la pagina;l'azione di pagina estrae i dati di input dalla

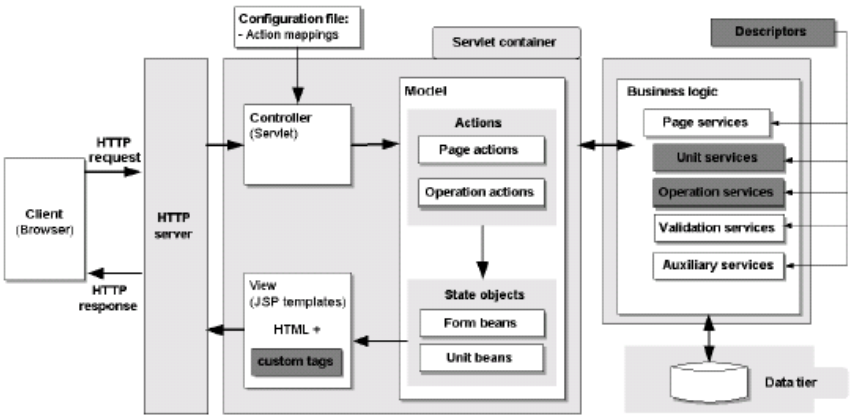

Figure 4.6: Architettura di un'applicazione WebRatio

richiesta HTTP ed invoca il servizio di pagina nel livello di business,passando i parametri necessari.

Il *servizio di pagina* è una funzione del livello di business che supporta la computazione del contenuto di una pagina invocata dall'azione di pagina.Alla fine dell'esecuzione del servizio di pagina,vengono resi disponibili per la Vista un'insieme di oggetti Java che memorizzano il contenuto delle unità della pagina(chiamate *unit beans*).

Il *template di pagina* è un template JSP,il quale elabora la pagina HTML da fornire all'utente,basata sul contenuto del Modello.
Infine,il *mapping di azione* è una dichiarazione nel file di configurazione del Controllore che collega la richiesta dell'utente,l'azione di pagina ed il template di pagina.

Ogni unit WebML viene mappata in due componenti:un *servizio di unit* nel livello di business ed un insieme di *custom tags* nella Vista.
Un *servizio di unit* è una classe Java responsabile dell'elaborazione

del contenuto dell'unità e della produzione di un insieme di *unit beans* che mantengono il contenuto della unit.La classe contiene le istruzioni per formulare la query di data retrivial,eseguirla e depositare i risultati in opportune *unit beans*.
Le content unit richiedono anche l'implementazione di *custom tags* usati nei templates di pagina della vista per trasformare in HTML il contenuto della unit memorizzato nelle unit beans.Tali tags possono essere presi da una libreria di tag JSP standard o definiti appositamente.
Le operation unit vengono invece mappate in tre componenti: un'*azione di operazione* nel Modello,simile all'azione di pagina,un *servizio di operazione* nel livello di business,simile ad un servizio di unit, ed un *mapping di azione* nel file di configurazione del Controllore che determina il flusso di controllo dopo il termine dell'esecuzione dell'operazione.

Il tipico flusso di controllo in seguito alla richiesta dell'utente al Web server è il seguente:

- La richiesta viene indirizzata al Controllore,implementato come un servlet Java;il Controllore consulta i mappings d'azione nel suo file di configurazione e decide quale azione di pagina invocare per la costruzione della pagina.
- Viene chiamata l'azione di pagina associata con la pagina,si tratta di una classe Java che interagisce con il livello di business.L'azione di pagina invoca il servizio di pagina associato alla pagina richiesta,il quale si occupa della costruzione della pagina.Inoltre l'azione di pagina deve anche memorizzare il contenuto della richiesta HTTP e della sessione JSP in opportuni oggetti Java,in modo che i servizi nel livello di business rimangano indipendenti dal front-end Web.
- Il servizio di pagina invoca la sequenza corretta di servizi associati con le units che compongono la pagina.Ogni servizio

di unit mantiene la logica di esecuzione della unit corrispondente che tipicamente consiste nell'esecuzione di queries per l'estrazione di dati.I risultati di queste queries vengono memorizzati nelle unit beans del Modello.

- In seguito il controllo ritorna al Controllore che consulta nuovamente i mappings d'azione per decidere quale template di pagina invocare per mostrare il contenuto delle unit beans.

- Viene costruita la pagina HTML da inviare all'utente,eseguendo il template di pagina appropriato.Tale esecuzione comporta anche l'esecuzione dei *custom tags*utilizzati per presentare il contenuto delle units della pagina.Alcuni di questi tags possono essere ancore o forms HTML,che permettono all'utente di inviare nuove richieste.

il flusso di controllo nel caso di invocazione di un'operazione risulta molto simile,ma con alcune differenze:

- La classe invocata dal Controllore è un'azione d'operazione,non un'azione di pagina.

- Nel livello di business ogni operazione corrisponde esattamente ad un solo servizio d'operazione,non ad un servizio di pagina o ad un insieme di servizi di unit.

- Quando il servizio di operazione termina ,ritorna il codice risultante nel unit bean;tale codice viene ripassato dall'azione dell'operazione al Controllore per prendere la decisione del link da seguire (OK o KO).

- Dopo l'esecuzione,se l'operazione è la i-sima di una sequenza,il Controllore invoca le operazioni seguenti oppure l'azione di pagina associata con la pagina puntata dal link di uscita(OK o KO),nel caso in cui l'operazione è l'ultima della sequenza.

È importante sottolineare che il codice di tutti i componenti descritti non deve essere scritto .
Infatti alcuni di tali componenti fanno parte della **Struttura di Runtime di WebRatio**:il Controllore,le azioni di pagina ed i servizi di pagina;altri componenti vengono invece creati automaticamente dal generatore di codice di WebRatio.
WebRatio genera i descrittori XML delle pagine,dei links,delle content ed operation units e dei JavaBeans usati per memorizzare i dati delle content unit.Inoltre vengono generati automaticamente i templates di pagina,il file di configurazione del Controllore,le classi Java delle azioni di pagina e d'operazione e le classi Java per le azioni di form della pagina per la validazione dell'input dell'utente.

Applicazione di WebRatio ad un caso pratico

Il CASE tool *WebRatio Site Development Studio* è stato utilizzato per lo sviluppo di applicazioni Web coinvolte nel sito Web della Facoltà di Ingegneria dell'Università di Modena e Reggio Emilia,le quali naturalmente erano già state realizzate con altri strumenti.
In particolare è stata sviluppata un'applicazione Web riguardante:

1. La gestione (inserimento,classificazione e ricerca) delle pubblicazioni effettuate dai docenti della facoltà.

2. La gestione dello spazio per ogni insegnamento contenente informazioni generali sull'insegnamento e il relativo materiale didattico.

3. La gestione delle pagine personali (home page) dei docenti della facoltà.

4. La gestione delle informazioni relative ai corsi di studi e alla loro organizzazione didattica:gli orientamenti di ogni

corso di studi,gli insegnamenti che compongono un corso di studi,l'assegnazione degli insegnamenti ai docenti,la composizione delle commissioni d'esame,le sessioni d'esame,gli appelli d'esame e le liste di studenti per ogni appello d'esame.

Nella creazione di tali applicazioni sono stati seguiti i passi principali del processo di sviluppo di WebRatio:

Specifica della struttura:

Il modello strutturale di WebRatio,come sottolineato nella descrizione del linguaggio WebML ,adotta la *notazione Entity-Relationship* ed è compatibile con i *diagrammi delle classi UML*. Gli elementi fondamentali del diagramma strutturale sono le entità con i loro attributi e le relazioni tra esse.

Creando un nuovo progetto ,nell'area di lavoro compare un diagramma strutturale di default,mostrato in figura 4.7,contenente tre entità predefinite: *User*,

Group e *SiteView* e le loro relazioni.Tali entità predefinite modellano la presenza di gruppi differenti di utenti ad ogniuna delle quali corrisponde una particolare vista di sito e servono dunque per la gestione degli accessi,che verrà trattata in seguito.

Ogni utente appartiene ad uno o più gruppi tra i quali viene indicato un gruppo di default,spiegando così la doppia relazione tra User e Group.

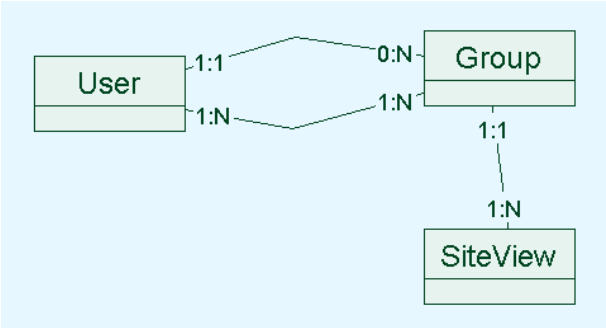

Figure 4.7: Le entità predefinite *User*,*Group* e *SiteView*

L'area di lavoro per la modellazione della struttura assiste l'utente nella creazione della struttura dei dati della propria applicazione mettendo a disposizione un'interfaccia grafica semplice ed intuitiva.Vengono inserite entità,attributi,relazioni e gerarchie semplicemente cliccando sulla relativa icona.
I comandi di zoom e di vista complessiva permettono di organizzare e gestire anche diagrammi strutturali complessi.
È possibile inoltre modificare l'aspetto del diagramma strutturale in vari modi,per esempio decidendo se mostrare o meno i nomi degli attributi delle entità o passando dalla notazione E/R a quella dei diagrammi delle classi UML per quanto riguarda l'indicazione delle cardinalità delle relazioni.
La figura 4.8mostra il diagramma strutturale complessivo delle applicazioni esaminate e l'area di lavoro dedicata con le icone dei vari strumenti.Si è scelto di mostrare gli attributi all'interno delle entità e di adottare la notazione E/R per le cardinalità.
In seguito vengono descritte le strutture dati utilizzate nel modello strutturale:

1. Per quanto riguarda la *gestione delle pubblicazioni*,si fa riferimento ad uno standard de facto delle pubblicazioni scientifiche quale è quello usato nell'applicazione BibTex,in base al quale le informazioni relative ad una pubblicazione variano a seconda del tipo di pubblicazione.
   Vengono considerati tre tipi principali di pubblicazioni:libro,in proceeding ed articolo ,rappresentate nel diagramma strutturale dalle entità *Pubblicazione Libro*, *Pubblicazione Articolo* e *Pubblicazione In Proceeding* con i loro rispettivi attributi.
   Ogni pubblicazione è caratterizzata dall'appartenere ad un'area di ricerca (entit à *AreadiRicerca*).
   L'entità *AreadiRicercaRapporto* rappresenta invece il rapporto annuale sulla ricerca svolta in facoltà in una certa

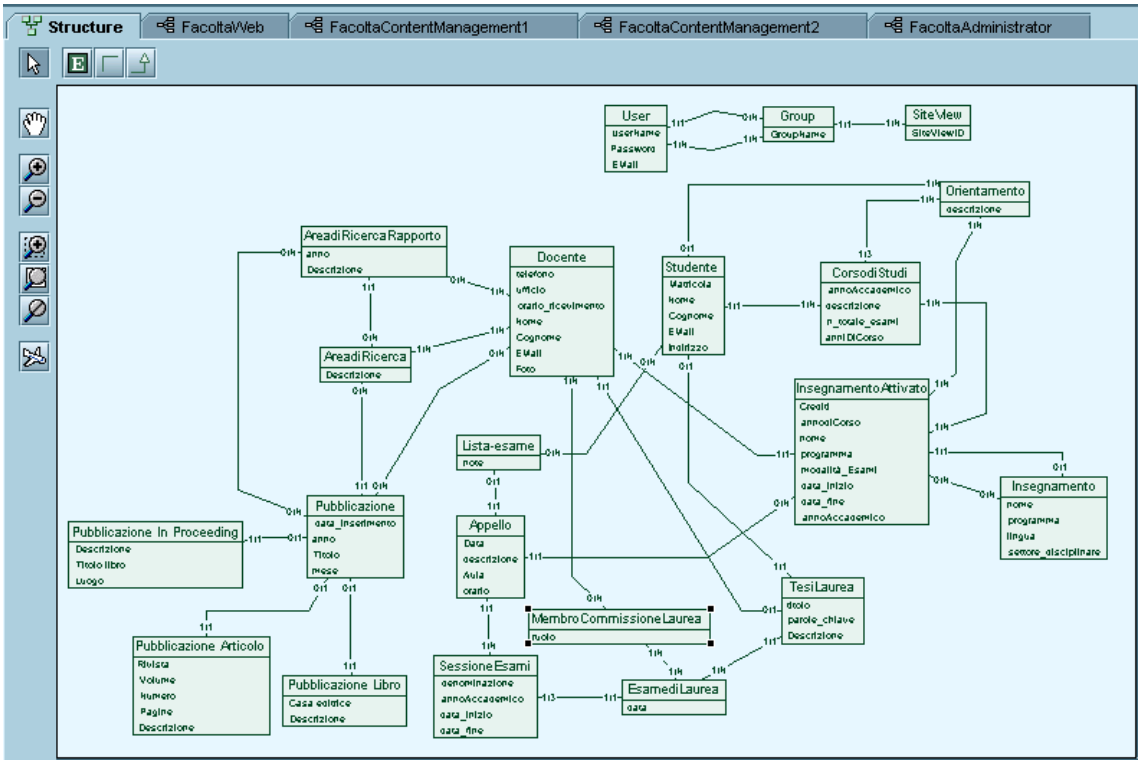

Figure 4.8: Modello strutturale complessivo

area ed è caratterizzata da una bibliografia espressa come insieme di pubblicazioni,oltre che da una descrizione,un docente responsabile,un anno ed un'area dui ricerca relativa. La relazione tra le entità *Docente* e *Pubblicazione*indica quali sono gli autori della pubblicazione o viceversa le pubblicazioni personali dei docenti.

2. Per quanto riguarda la *gestione dello spazio per ogni insegnamento* vengono visualizzate solo le informazioni relative agli insegnamenti attivati in un determinato anno accademico.Infatti,come si può vedere dal diagramma strutturale,vengono considerate due entità distinte:*Insegnamento* ed *InsegnamentoAttivato* per sottolineare l'attivazione di un insegnamento;inoltre viene rappresentata un'ulteriore

relazione tra le due entità per rappresentare la propedeuticità.
Ogni insegnamento attivato è caratterizzato ,oltre che dai propri attributi,da un docente titolare ,da un certo numero di appelli e da un determinato corso di studi e/o orientamento a cui appartiene.

3. Per quanto riguarda la *gestione delle home pages dei docenti*,a livello strutturale le home pages non vengono rappresentate in quanto non sono altro che la visualizzazione di daterminati attributi e relazioni dell'entità *Docente*.Tuttavia tale entità presenta le relazioni necessarie per i links che dovranno apparire nelle home pages:gli insegnamenti tenuti dal docente,le aree di ricerca di appartenenza e le pubblicazioni personali del docente.

4. Per quanto riguarda la *gestione dell'organizzazione didattica*,le informazioni relative ai corsi di studio vengono rappresentate con l'entità *CorsodiStudi*,caratterizzata da un anno accademico in cui il corso è attivo,da una descrizione,da un numero totale di esami e dagli anni di corso.Inoltre ogni corso di studi è caratterizzato dagli studenti iscritti ,dagli esami di base e dai possibili orientamenti,rappresentati dall'entità *Orientamento* che,analogamente a *CorsodiStudi*,è in relazione con gli studenti iscritti all'orientamento e con gli insegnamenti di orientamento.
Per quanto rigurda gli esami vengono considerate le sessioni d'esame (entità *SessioneEsami*) all'interno delle quali vengono definiti appelli (*Appello*) ed esami di laurea (*Esamedi-Laurea*).
Per rappresentare l'iscrizione di uno studente ad un appello è stata creata un'entità *Lista-esame*,la quale mantiene l'informazione dello studente iscritto e dell'appello permettendo di specificare alcune note d'iscrizione.Ogni esame di

laurea è caratterizzato da un insieme di tesi discusse (entità*TesiLaurea*),collegate con lo studente autore ed il docente relatore, e da una commissione di laurea composta da docenti.Analogamente al caso delle note di iscrizione,per permettere di specificare il ruolo di ogni docente nella commissione di laurea è stata creata l'entità *MembroCommissioneLaurea*.

Specifica delle site views:
Una *site view*,come descritto nel paragrafo 3.1.3,rappresenta un particolare ipertesto associato ad uno o più gruppi di utenti ed offre un'interfaccia grafica per la modellazione della composizione delle pagine e della navigazione.
Per quanto rigurda l'applicazione sviluppata che realizza alcune funzionalità del sito web della Facoltà di Ingegneria,per semplicità si considerano tre gruppi di utenti:

- Studenti
- Docenti
- Amministratore

Ad ogni gruppo viene associata una particolare site view sulla base del diagramma strutturale di default di figura 4.7 solo l'amministratore appartiene a tutti i gruppi ed ha una visione globale dell'applicazione.
Proprio in base ai diritti di accesso le site views possono essere:

- Site Views pubbliche:possono essere aperte da chiunque,non richiedono un login.
- Site Views private:sono accessibili solo da utenti selezionati ,i quali devono specificare username e password per visualizzare le pagine della site view.

Nell'applicazione sviluppata sono state definite quattro site views:

- Una site view pubblica:**FacoltaWeb**.
- Tre site views private,corrispondenti ai tre gruppi di utenti:
  - Una site view per gli studenti:**FacoltaContentManagement1**.
  - Una site view per i docenti:**FacoltaContentManagement2**.
  - Una site view per l'amministratore:**FacoltaAdministrator**.

Ogni site view è associata ad un pannello dedicato dell'area di lavoro ed è formata da pagine raggruppate in aree.
Ogni site view è caratterizzata da una home page;inserendo tale pagina è necessario specificare la proprietà di *Home* raffigurata con una H in alto a destra.
Per ogni area deve essere definita una pagina di default,distinta da una D in basso a destra.
Inserendo una pagina o un'area è possibile specificare o meno la proprietà di *Landmark*.Una pagina/area con tale proprietà è raggiungibile da tutte le pagine/aree della site view e presenta una L in basso a destra.
In particolare ,le aree landmark vengono solitamente visualizzate nella parte alta del sito e rappresentano le categorie di servizi/informazioni a cui è sempre possibile accedere.

La site view FacoltaWeb:
Come specificato, la site view FacoltaWeb è una vista pubblica a cui chiunque può accedere per consultare informazioni relative a:Corsi di Studi,Sessioni d'esame,Insegnamenti,Docenti,Aree di Ricerca e Pubblicazioni.Tali categorie di informazioni rappresentano le aree landmark della site view.
In figura 4.9 sono mostrate le proprietà della site view.

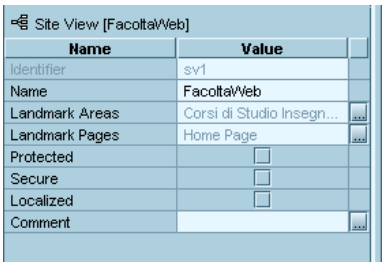

Figure 4.9: Proprietà della site view FacoltaWeb

La figura 4.10 mostra una visione globale della site view allo scopo di visualizzare la disposizione di pagine ed aree;naturalmente l'immagine risulta poco significativa,ma ciascuna area verrà mostrata ed analizzata a parte.

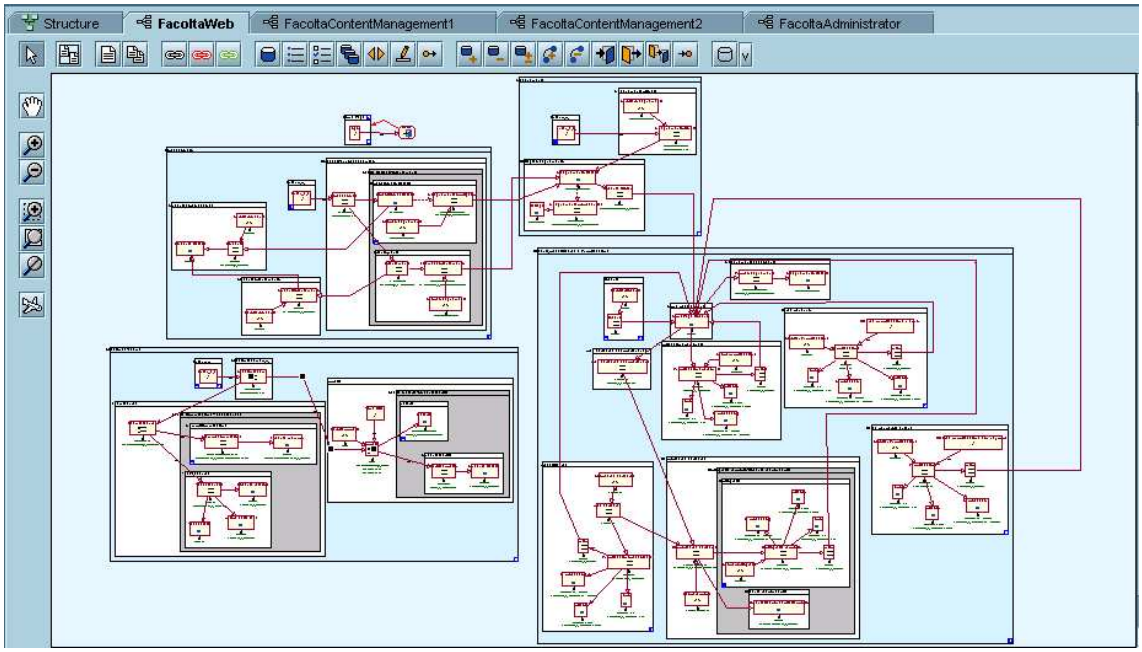

Figure 4.10: Visione globale della site view FacoltaWeb

L'area *Corsi di Studio*,in figura 4.11,permette di visualizzare i corsi di studio attivi nell'anno accademico di interesse che viene richiesto all'utente nella entry unit della home page.Per ogni corso è possibile conoscere gli insegnamenti obbligatori ,gli studenti iscritti ed i possibili orientamenti con i relativi esami di orientamento e studenti iscritti.
In quest'area è interessante l'uso delle pagine alternative,chiamate *Scelta Dettagli/Orientamento*, che permette di visualizzare una i dettagli del corso di studi scelto piuttosto che i relativi orientamenti a seconda dell'interazione dell'utente.Da notare che la pagina con i dettagli del corso presenta una D in basso a destra che la indica come pagina di default dell'alternativa.

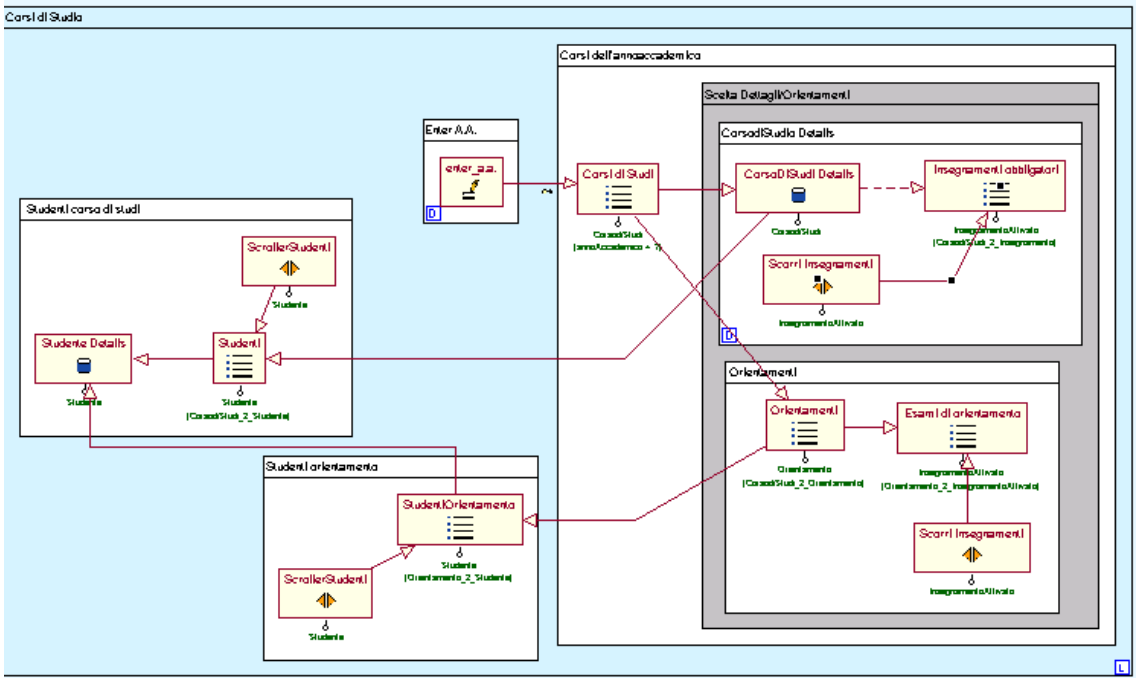

Figure 4.11: Area *Corsi di Studio* della site view *FacoltaWeb*

L'area *Insegnamenti*,in figura 4.12,mostra gli insegnamenti attivati per un certo anno accademico immesso dall'utente,analogamente ai corsi di studio.Scelto un'insegnamento ne vengono mostrate le caratteristiche specifiche,il docente titolare e gli insegnamenti propedeutici,i quali possono essere anche insegnamenti non attivati.

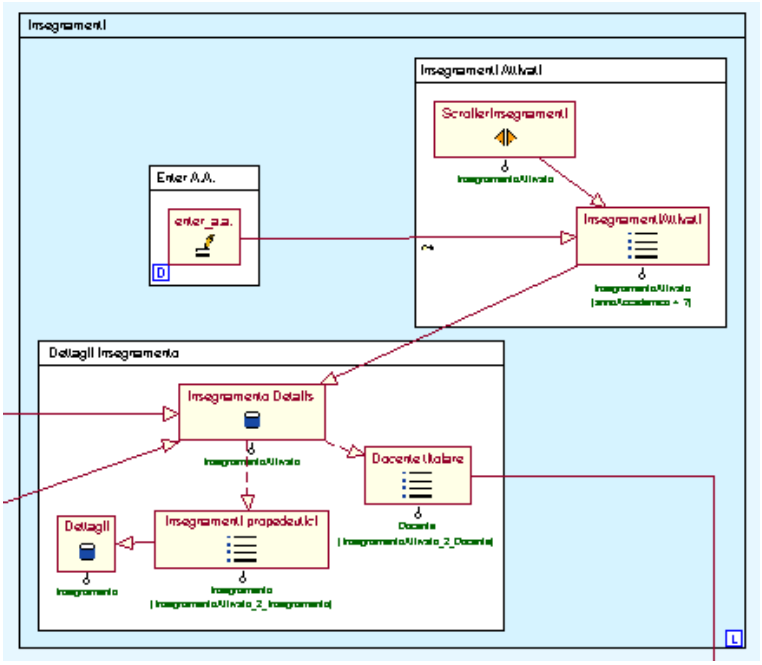

Figure 4.12: Area *Insegnamenti* della site view *FacoltaWeb*

L'area *Sessioni d'Esame*,in figura 4.13,permette di visualizzare le sessioni d'esame dell'anno accademico inserito dall'utente;una volta scelta la sessione d'interesse si possono visitare la pagina relativa agli appelli della sessione o la pagina relativa agli esami di laurea della sessione.
Nella pagina degli appelli ,l'utente deve inserire una data minima ed una massima allo scopo di creare un filtro di tempo per gli appelli da visualizzare;per ogni appello è possibile visualizzare,in alternativa,la materia e gli studenti iscritti.
Nella pagina degli esami di laurea,scelto un esame,vengono mostrate le informazioni relative alla commissione di laurea oppure le tesi discusse.

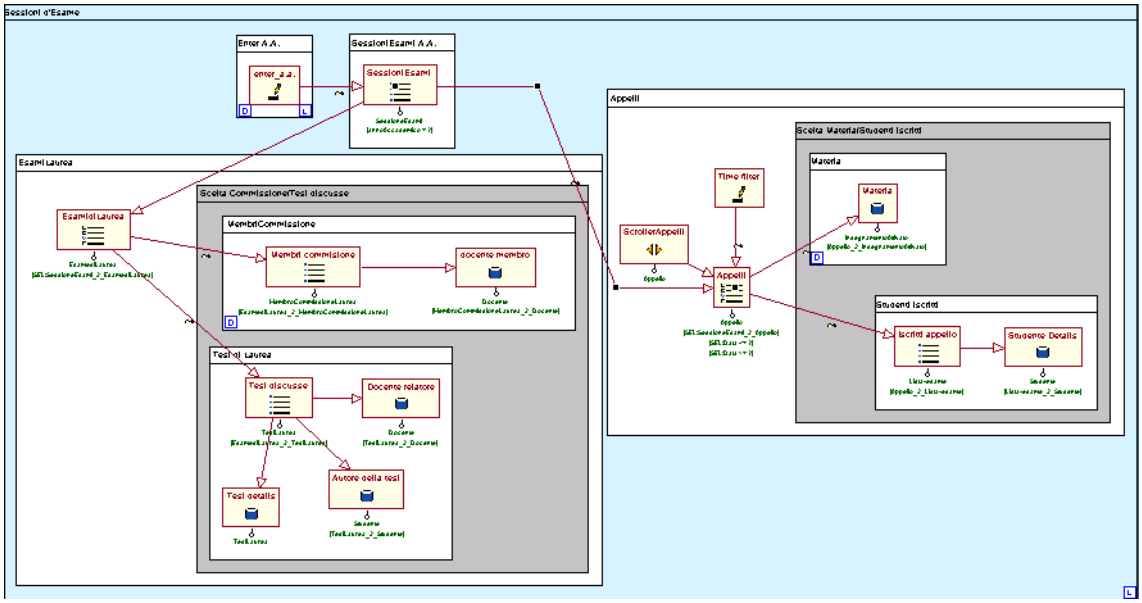

Figure 4.13: Area *Sessioni d'Esame* della site view *FacoltaWeb*

L'area *Docenti,AreediRicerca e Pubblicazioni* rappresenta l'area più complessa della site view in quanto racchiude la gestione delle home pages dei docenti,delle pubblicazioni e delle aree di ricerca.
Dopo aver scelto il docente d'interessa,l'home page relativa,come mostrato nella porzione della site view in figura 4.14,permette di accedere agli insegnamenti del docente,alle aree di ricerca di appartenenza ed alle pubblicazioni personali del docente.

La porzione relativa alle aree di ricerca e ai rapporti di tali aree(figura 4.15),scelta un'area di ricerca,ne mostra le pubblicazioni ed i rapporti.
Per ogni rapporto si può visualizzare la bibliografia relativa oppure il docente responsabile.
Da notare che ogni pubblicazione può essere di un solo tipo;dunque le tre data units *Libro*, *Articolo* ed *In proceeding* non vengono mai visualizzate insieme poichè solo una di esse può contere dati

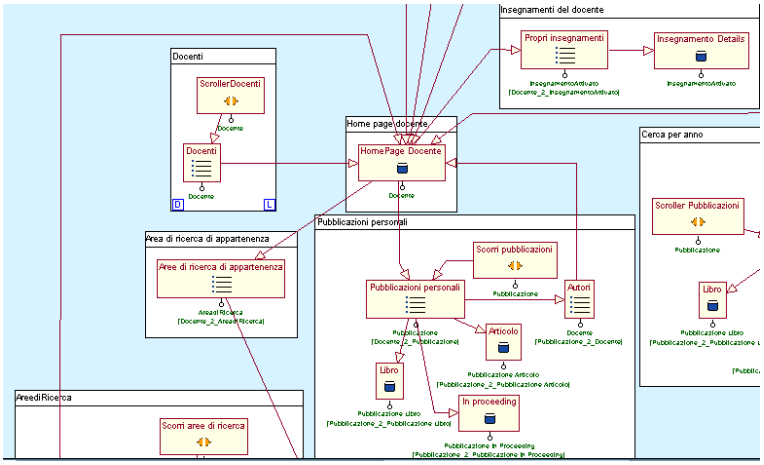

Figure 4.14: Porzione dell'area *Docenti,AreediRicerca* e *Pubblicazioni* riguardante le home pages dei docenti

per una certa pubblicazione.

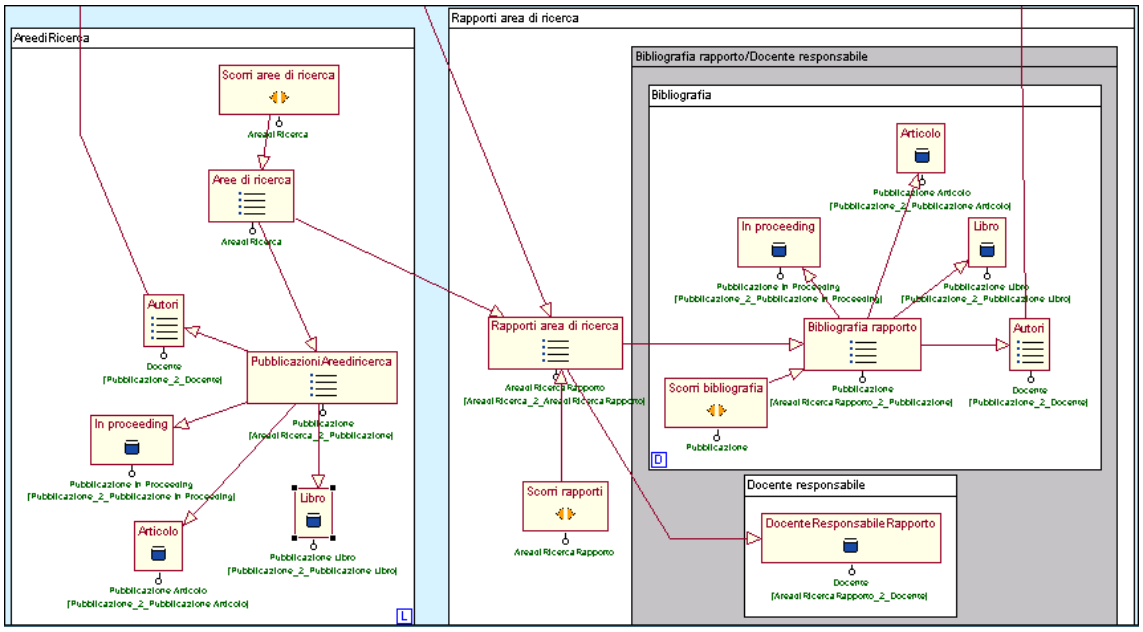

Figure 4.15: Porzione dell'area *Docenti,AreediRicerca* e *Pubblicazioni* riguardante le aree di ricerca

Per quanto riguarda le pubblicazioni è possibile effettuare una ricerca per anno (figura 4.16) o in base a parole chiave(figura 4.17).

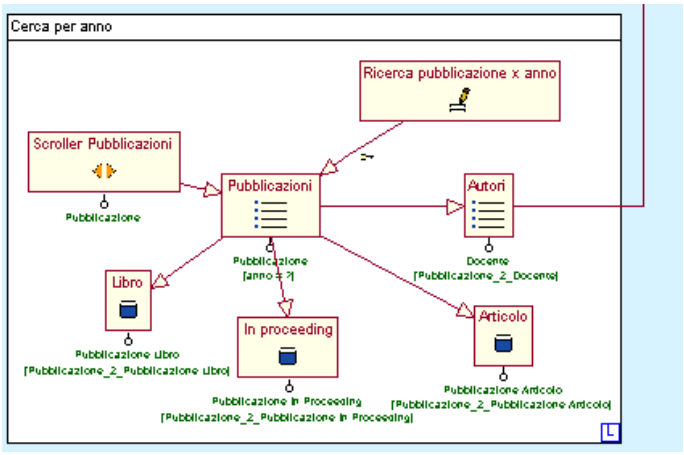

Figure 4.16: Porzione dell'area *Docenti,AreediRicerca* e *Pubblicazioni* riguardante la ricerca di pubblicazioni per anno

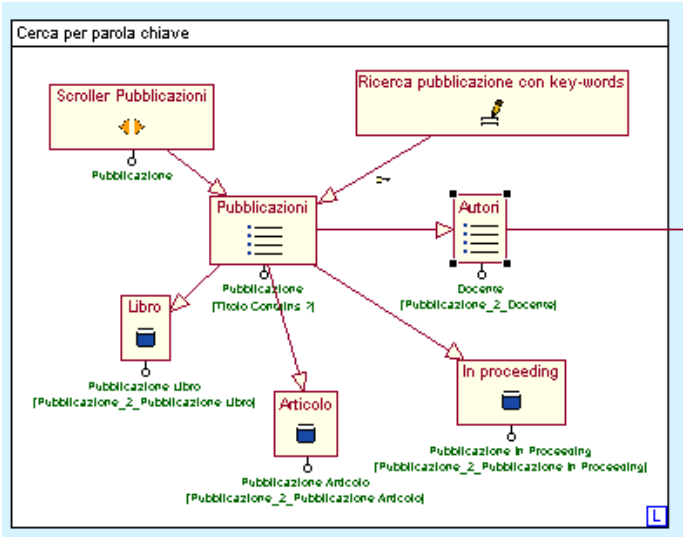

Figure 4.17: Porzione dell'area *Docenti,AreediRicerca* e *Pubblicazioni* riguardante la ricerca di pubblicazioni per parole chiave

La site view FacoltaWeb,come tutte le site view pubbliche,contiene un'entry unit per permettere all'utente di inserire username e password ed effettuare il login alle site views private mediante la *login unit*.
Nella site view FacoltaWeb la form per il login è posizionata nella home page,come mostra la figura 4.18.

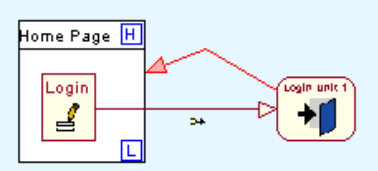

Figure 4.18: Home page della site view *FacoltaWeb* con entry unit per il login

La site view FacoltaContentManagement1:
Come specificato,la site view FacoltaContentManagement1 è riservata agli utenti che appartengono al gruppo degli studenti.
Si tratta di una site view in cui non è solo possibile visualizzare informazioni,ma anche effettuare operazioni su tali informazioni ,come la modifica,cancellazione o inserimento.
Le operazioni permesse all'utente studente sono:

- Visualizzare le proprie iscrizioni agli esami a partire da una data inserita dall'utente,modificare tali iscrizioni,cancellarle o inserire una nuova iscrizione.
- Modificare,cancellare o inserire le informazioni relative alla propria tesi di laurea.

Le aree landmark della site views rispecchiano queste operazioni,infatti sono *PROPRIA TESI* e *PROPRI ESAMI*.

In figura 4.19 sono mostrate le proprietà della site view.

La figura 4.20 mostra una visione globale della site view allo scopo di visualizzare la disposizione di pagine ed aree;naturalmente

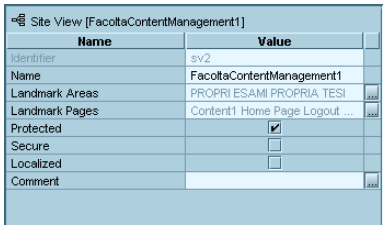

Figure 4.19: Proprietà della site view *FacoltaContentManagement1*

l'immagine risulta poco significativa,ma ciascuna area verrà mostrata ed analizzata a parte.

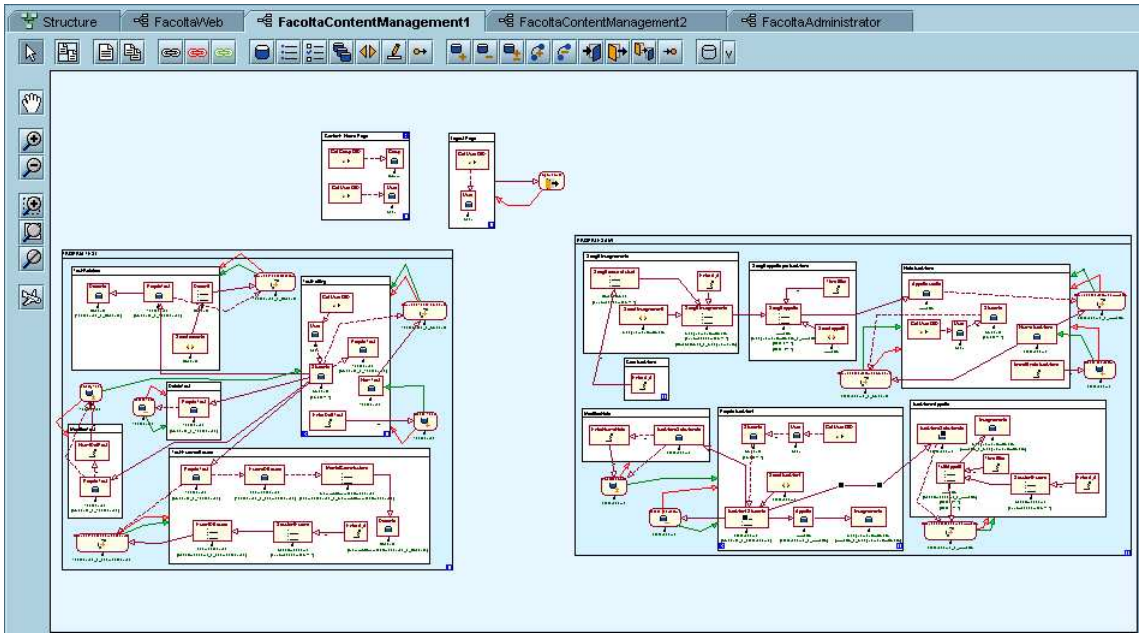

Figure 4.20: Visione globale della site view *FacoltaContentManagement1*

Nell'area *PROPRI ESAMI*,in figura 4.21, sono presenti due pagine con la proprietà landmark:la pagina di default *Proprie Iscrizioni* e la pagina *Crea Iscrizione*;infatti lo studente può visualizzare tutte le proprie iscrizioni agli esami oppure iscriversi per un nuovo esame.
Nella creazione di una nuova iscrizione ,lo studente inizialmente deve scegliere l'insegnamento del quale vuole sostenere l'esame tra quelli attivati per l'anno accademico desiderato,in seguito deve essere scegliere l'appello a cui iscriversi,
specificando una fascia temporale, ed immettere note relative all'iscrizione(ad esempio: solo primo scritto,solo orale etc.).
Per quanto riguarda le proprie iscrizioni già effettuate,lo studente,oltre a visualizzarle,può cancellare un'iscrizione oppure modificarne l'appello corrispondente o le note.

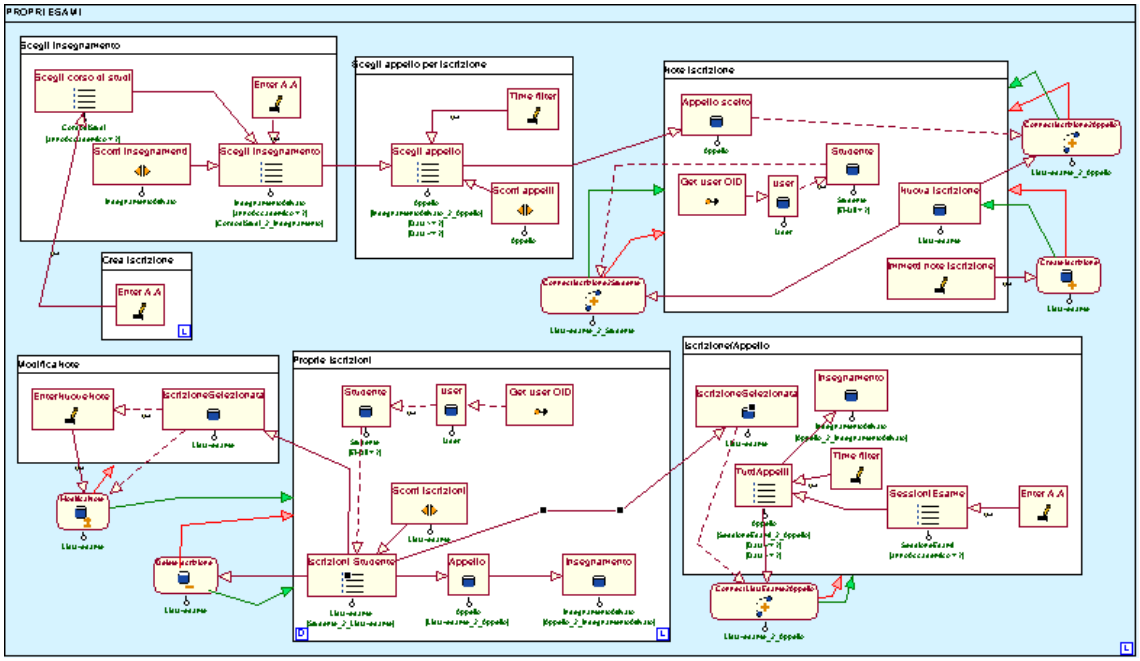

Figure 4.21: Area *PROPRI ESAMI*della site view *FacoltaContentManagement1*

Nell'area *PROPRIA TESI*,in figura 4.22,lo studente può visualizzare i dati relativi alla propria tesi,cancellarla o modificarla:possono essere inseriti nuovi dati(descrizione,parole chiave,titolo),si può cambiare il relativo esame di laurea o il docente relatore.
Inoltre lo studente può inserire la propria tesi,immettendo i dati relativi,l'esame di laurea in cui la tesi è stata discussa ed il docente relatore.

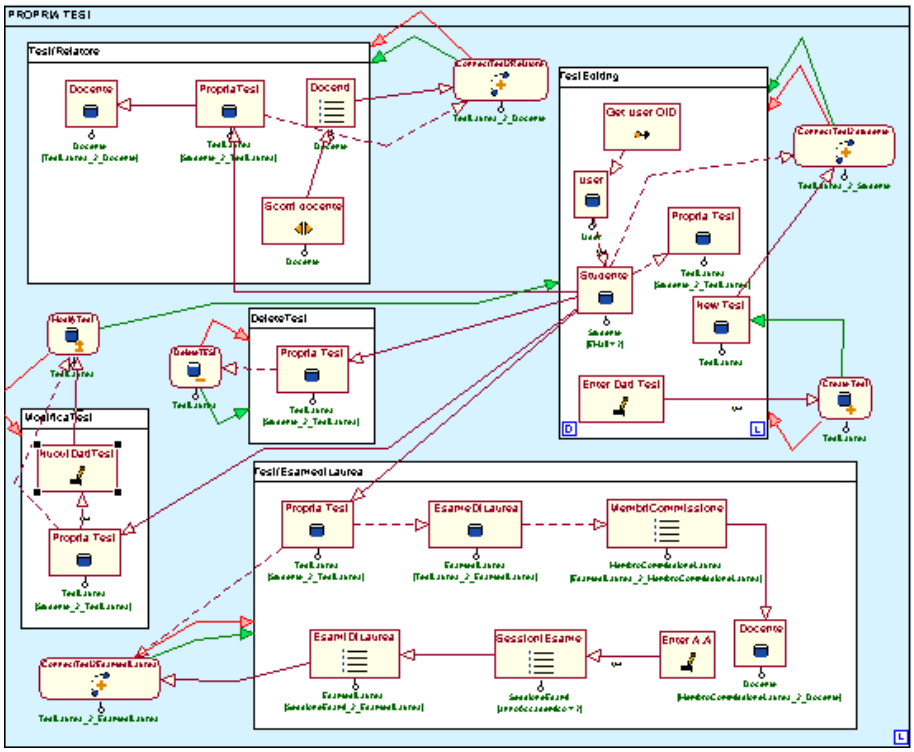

Figure 4.22: Area *PROPRIA TESI* della site view *FacoltaContentManagement1*

La home page della site view *FacoltaContentManagement1*,in figura 4.23, contiene un elemento di ipertesto, specifico delle site views private,che permette di visualizzare l'identità dell'utente che ha effettuato il login ed i dati relativi al gruppo associato a tale site view.
Ciò viene realizzato per mezzo di due *Get Units* basate su due parametri globali predefiniti in WebRatio per mantenere traccia dell'attuale utente e del relativo gruppo:*GROUPCTXPARAM* e *USERCTXPARAM*.
In ogni site view privata deve inoltre essere presente una pagina di logout per la chiusura della sessione dell'utente e l'apertura della home page della site view pubblica dell'applicazione.
La pagina di logout della site view *FacoltaContentManagement1*

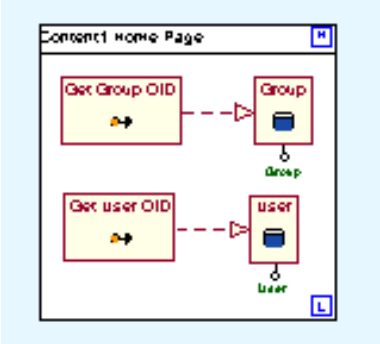

Figure 4.23: Home Page della site view *FacoltaContentManagement1*

,analoga per tutte le tre site views private,è mostrata in figura 4.24.Le due get e data units sono opzionali,vengono usate per visualizzare l'identità dell'utente che sta effettuando il logout.

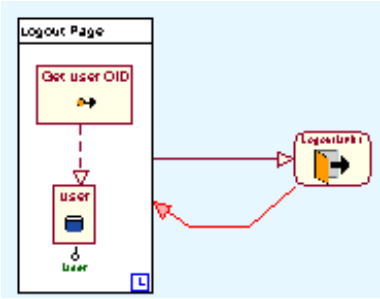

Figure 4.24: Logout Page della site view *FacoltaContentManagement1*

Il logout viene effettuato mediante una unit particolare,la *Logout Unit*,nelle cui proprietà,mostrate in figura 4.25,è necessario indicare la site view pubblica (in questo caso,FacoltaWeb) nella quale si desidera che l'utente si ritrovi dopo l'operazione di logout.

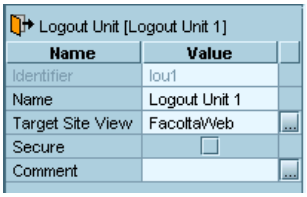

Figure 4.25: Proprietà della logout unit

La site view FacoltaContentManagement2:
Come specificato,la site view FacoltaContentManagement2 è riservata agli utenti che appartengono al gruppo dei docenti.
Analogamente alla site view FacoltaContentManagement1,non è solo possibile visualizzare informazioni,ma anche effettuare operazioni su tali informazioni ,come la modifica,cancellazione o inserimento.
Le operazioni permesse all'utente docente sono:

- Visualizzare i rapporti di aree di ricerca dei quali è responsabile,modificarli e cancellarli oppure inserire un nuovo rapporto di area di ricerca.
- Visualizzare gli appelli dei propri insegnamenti,modificarli e cancellarli oppure inserire un nuovo appello per un certo insegnamento.
- Visualizzare i dati delle proprie pubblicazioni personali,modificarle o cancellarle oppure inserire una nuova pubblicazione.

Tali operazioni rappresentano le aree di landmark della site view:*AREE DI RICERCA*,*APPELLI*,*PUBBLICAZIONI*.

In figura 4.26 sono mostrate le proprietà della site view.

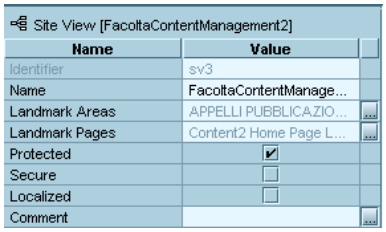

Figure 4.26: Proprietà della site view *FacoltaContentManagement2*

La figura 4.27 mostra una visione globale della site view allo scopo di visualizzare la disposizione di pagine ed aree;naturalmente

l'immagine risulta poco significativa,ma ciascuna area verrà mostrata ed analizzata a parte.

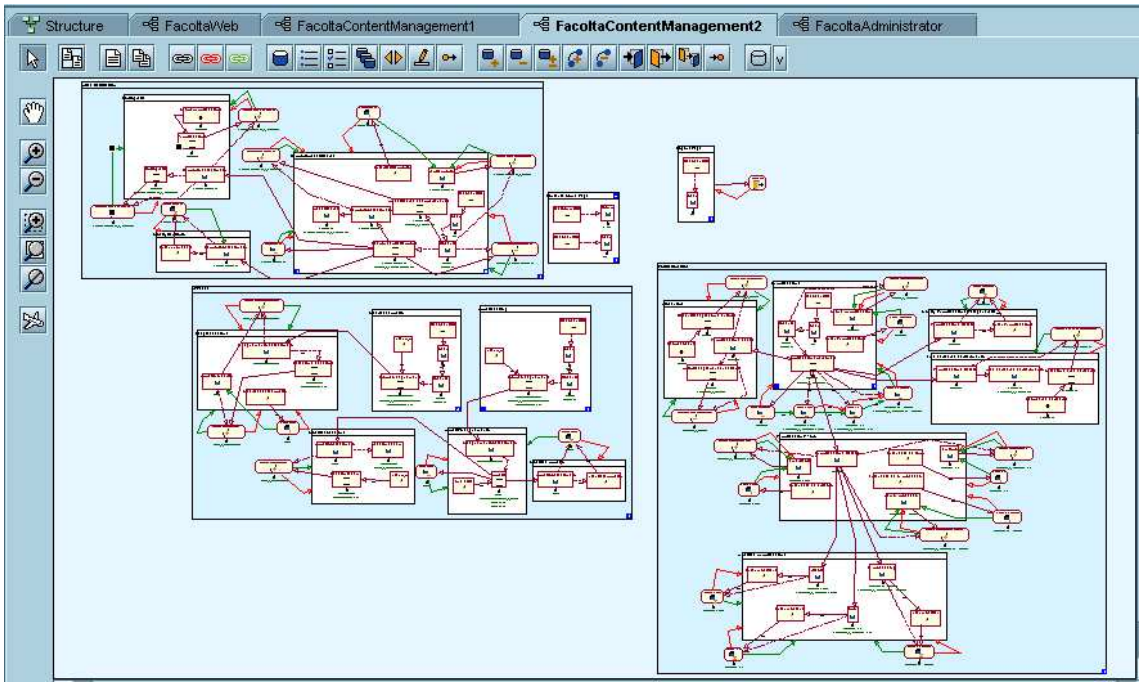

Figure 4.27: Visione globale della site view *FacoltaContentManagement2*

L'area *AREE DI RICERCA*,in figura 4.28 ,permette al docente di visualizzare i dati dei rapporti di area di ricerca dei quali è responsabile.Per ogni rapporto il docente può visualizzare e modifare la relativa bibliografia o area di ricerca , modificare i dati del rapporto(anno,descrizione) oppure eliminare la propria responsabilità in merito a tale rapporto,mediante l'uso della disconnect unit *Disconnect Responsabile2Rapporto*.
L'area considerata offre inoltre al docente la possibilità di inserire nuovi rapporti relativi alle aree di ricerca di appartenenza dei quali verrà considerato il responsabile.

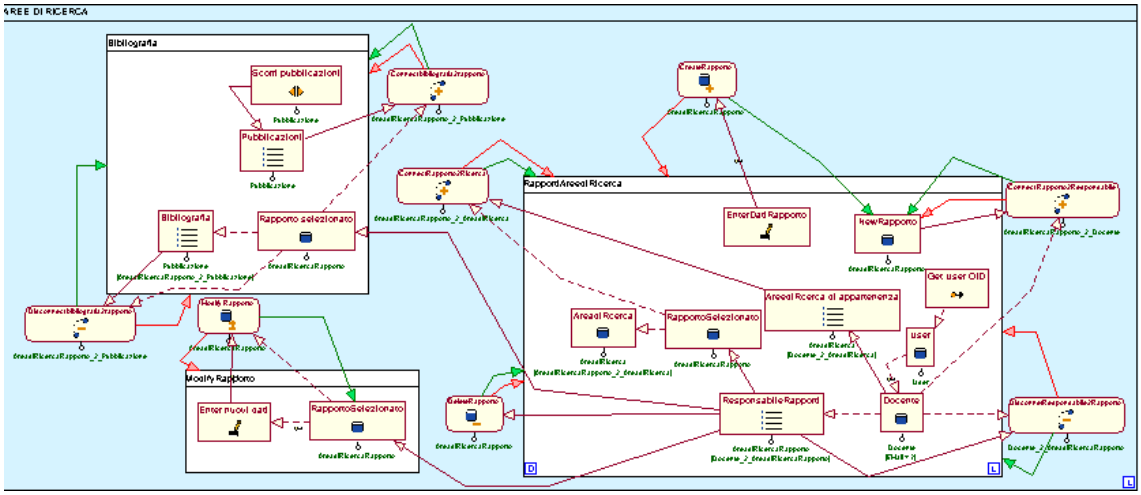

Figure 4.28: Area *AREE DI RICERCA* della site view *FacoltaContentManagement2*

L'area *APPELLI*,in figura 4.29,oltre alla pagina di default *Appelli Editing*,è presente anche un'altra pagina di landmark per l'inserimento di un nuovo appello:*Inserisci appello*.
Per quanto riguarda la creazione di un nuovo appello,il docente inizialmente deve scegliere per quale dei propri insegnamenti vuole inserire l'appello e la sessione d'esame relativa,in seguito devono essere inseriti i dati dell'appello:da
ta,aula,orario e descrizione.
Per quanto rigurda gli appelli già inseriti,è possibile visualizzarli inserendo una fascia temporale di limitazione, cancellarli oppure modificarne la sessione relativa o i dati.

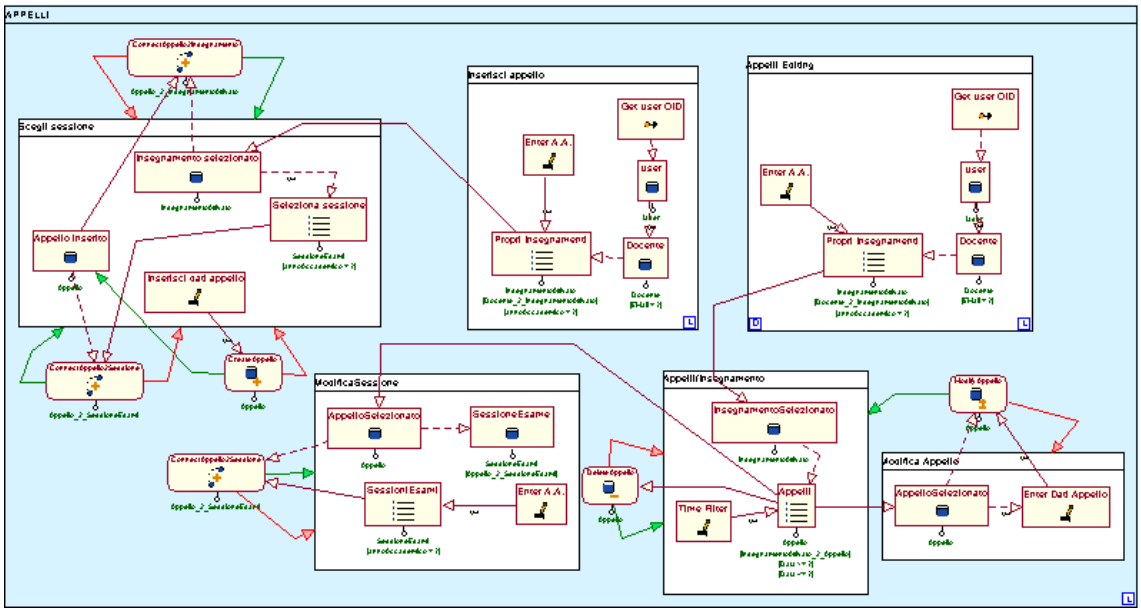

Figure 4.29: Area *APPELLI* della site view *FacoltaContentManagement2*

L'area *PUBBLICAZIONI* risulta piuttosto complicata a causa della scelta di utilizzare la notazione BibTex in base alla quale ad ogni tipo di pubblicazione corrispondono determinate proprietà.
Come mostrato nella porzione dell'area di figura 4.30 ,nella pagina di default l'utente-docente può inserire i dati per una nuova pubblicazione inoltre vengono mostrate al docente le proprie pubblicazioni.
Scelta una di tali pubblicazioni il docente ,limitatamente a quanto mostrato in figura 4.30,può:

- Cancellare la pubblicazione:dalla figura si può notare la cancellazione a cascata in base al tipo di pubblicazione.
- Visualizzare e modificare gli autori della pubblicazione:possono essere eliminati alcuni degli autori presenti o aggiunti nuovi autori.Si è supposto che gli autori delle pubblicazioni possano essere solo i docenti della facoltà.

- Visualizzare e modificare l'area di ricerca di riferimento della pubblicazione.
- Modificare i dati generali della pubblicazione:anno,mese,data di inserimento e titolo.

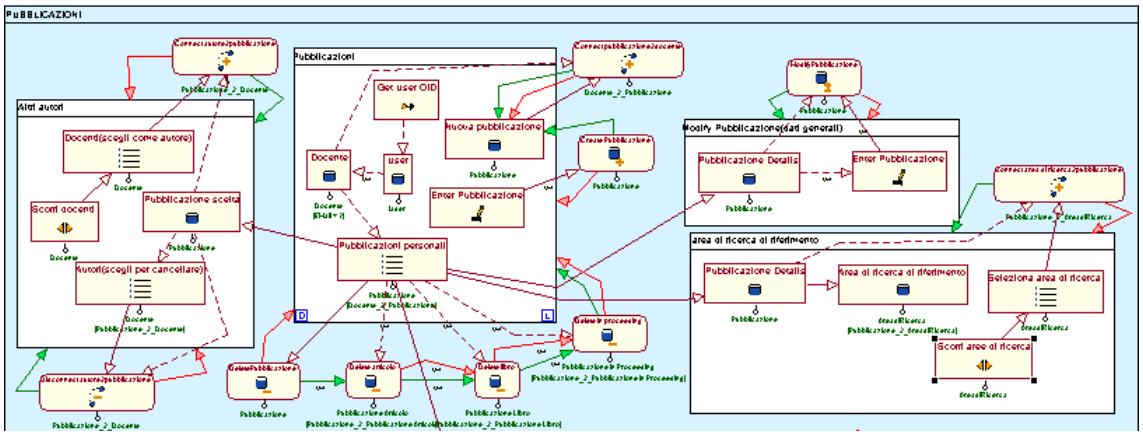

Figure 4.30: Porzione dell'area *PUBBLICAZIONI* della site view *Facolta-ContentManagement2*

Come mostrato nella porzione dell'area di figura 4.31,nell'ambito della creazione di una nuova pubblicazione i vari tipi di pubblicazione vengono gestiti chiedendo al docente di inserire dati diversi a seconda che la pubblicazione da inserire sia un articolo,un libro o un in proceeding.
Nella pagina *modifica publicazione* vengono mostrati i dati dettagliati di una pubblicazione in base al tipo ed è possibile modificare tali dati.

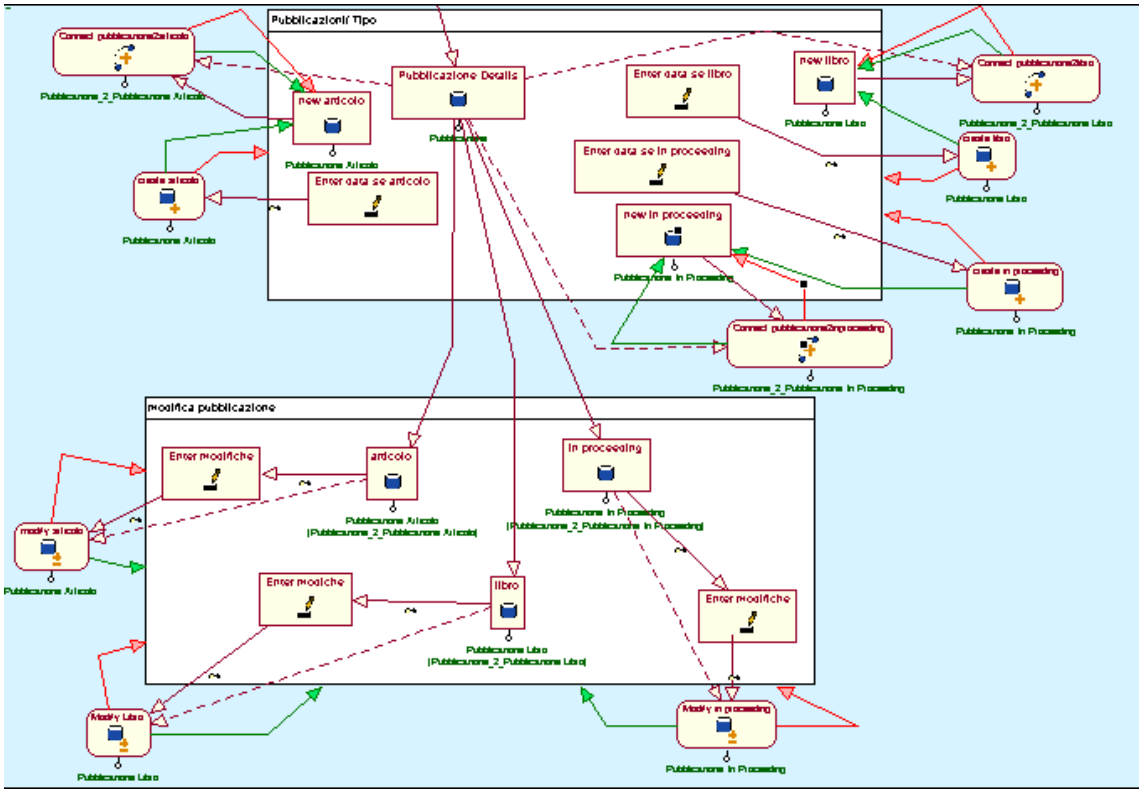

Figure 4.31: Porzione dell'area *PUBBLICAZIONI* della site view *Facolta-ContentManagement2*

La site view FacoltaAdministrator:
Nella site view *FacoltaAdministrator*,riservata all'amministratore,tutte le operazioni di creazione,cancellazione,modifica,connessione e disconnessione descritte nelle altre site views vengono applicate alla gestione dei profili utente.La gestione degli utenti consiste nel:

- Creare nuovi utenti.
- Modificare gli utenti esistenti,in termini di username e password.
- Cancellare gli utenti esistenti.
- Mettere in relazione utenti e gruppi.

In figura 4.32 sono mostrate le proprietà della site view.

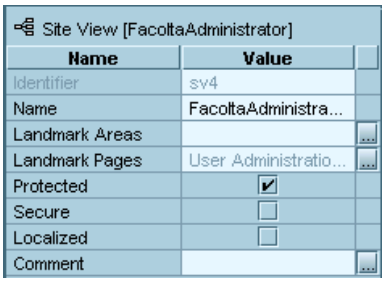

Figure 4.32: Proprietà della site view *FacoltaAdministrator*

La figura 4.33 mostra una visione globale della site view allo scopo di visualizzare la disposizione di pagine ed aree;naturalmente l'immagine risulta poco significativa,ma si può notare che tale site views non presenta una suddivisione in aree,ma solo pagine.

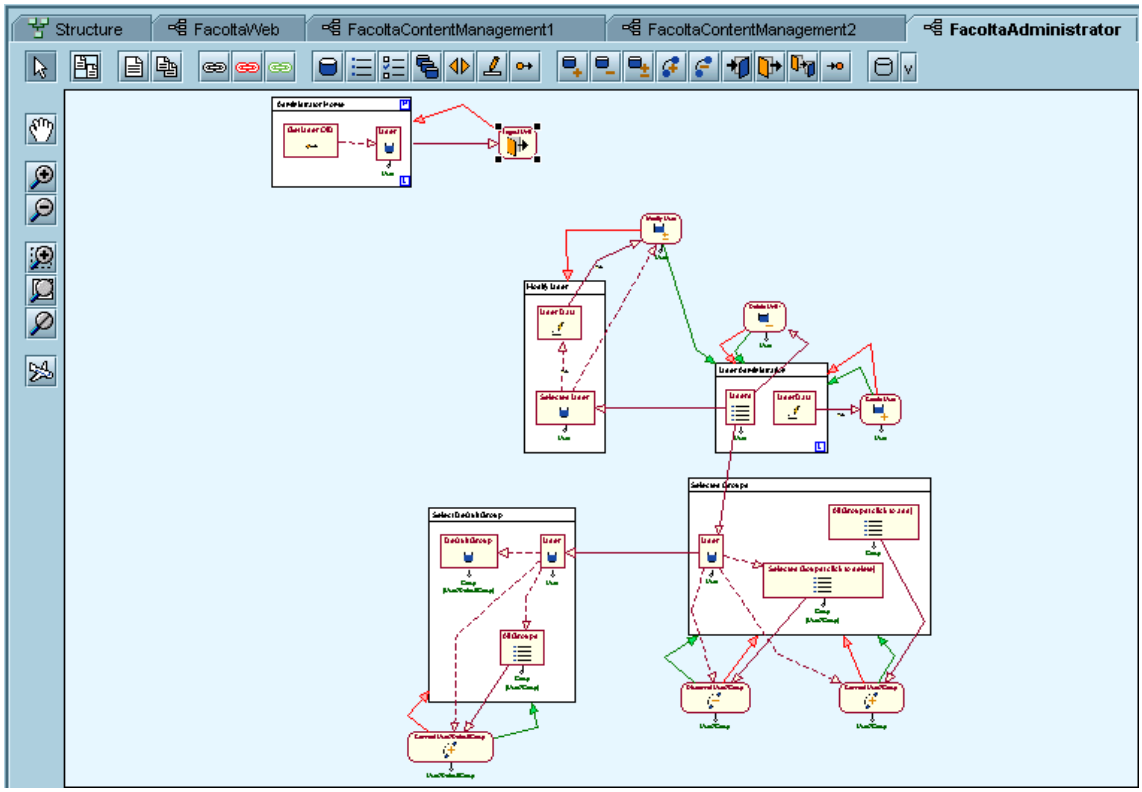

Figure 4.33: Visione globale della site view *FacoltaAdmin*

In figura 4.34 vengono mostrate le pagine per la creazione,cancellazione e modifica degli utenti,relizzate in modo analogo alle operazioni di questo tipo delle altre site views.

In figura 4.35 vengono mostrate le pagine rigurdanti le relazioni tra utenti e gruppi.

Il link in ingresso della pagina *Select Groups* trasporta come parametro l'identificatore dell'utente selezionato,che viene utilizzato per mostrare i gruppi ai quali l'utente appartiene,insieme ad una lista di tutti i gruppi disponibili.L'amministratore può realizzare la connessione utente-gruppo cliccando sul gruppo che vuole associare all'utente.Al contrario,cliccando su un gruppo a cui l'utente appartiene già ,viene realizzata un'operazione di disconnessione.

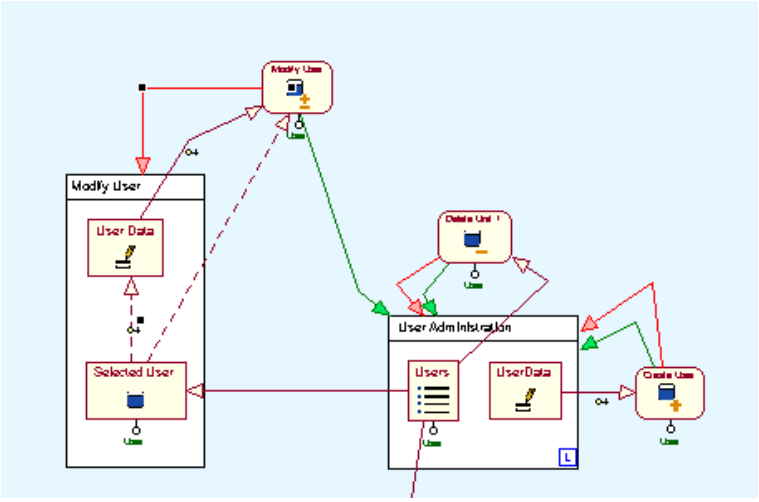

Figure 4.34: Porzione della site view *FacoltaAdmin* riguardante la gestione degli utenti

Per completare la definizione di un nuovo utente si deve definire il suo gruppo di default:nella pagina *Select Default Group* viene fornita una lista dei gruppi disponibili tra i quali è possibile scegliere il gruppo di default e collegarlo con l'utente.

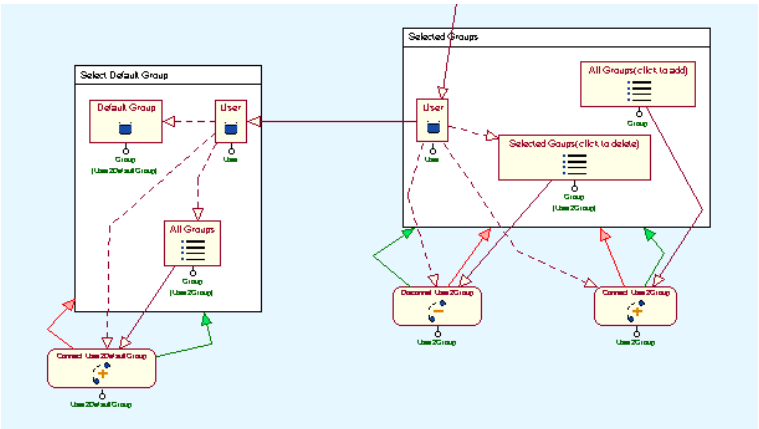

Figure 4.35: Porzione della site view *FacoltaAdmin* riguardante la gestione delle relazioni tra utenti e gruppi

Naturalmente anche la site view *FacoltaAdmin* contiene una pagina di logout analoga a quelle delle altre due site views private,in questo caso tale pagina rappresenta anche la home page della site view.

Terminata la modellazione dell'ipertesto,è stato effettuato il controllo degli errori selezionando il bottone *Structure-Navigation Warnings*(figura 4.36) dalla barra principale degli strumenti.

Figure 4.36: Bottone per i warning di struttura-navigazione

Creazione del database di supporto per l'applicazione:
Dopo aver definito la struttura e le site views dell'applicazione , è necessario creare un database ed alcuni dati di prova per simulare il contenuto dell'applicazione.
Il processo di definizione di un'origine di dati che mantenga il contenuto di un'applicazione WebRatio è chiamato *mapping*.
Un mapping associa gli elementi strutturali di WebML(entità,attributi e relazioni) ad una o più sorgenti di dati dove i dati attuali sono memorizzati.Il processo di mapping di un progetto può avvenire in due contesti diversi:

- Esiste già un database per il contenuto dell'applicazione:in questo caso lo schema stutturale WebML deve essere semplicemente mappato sulle tabelle e le viste dell'origine di dati.

- Non esiste un database per il contenuto dell'applicazione:in questo caso WebRatio può essere usato per creare un database di default contenente dati di prova e per mappare le entità e le relazioni su tale database.

Nello sviluppo dell'applicazione web rigurdante la facoltà,si assume che non esista nessun database allo scopo di utilizzare WebRatio per creare un database relazionale ed inserirvi dei dati di prova.
Innanzitutto è stato creato un database vuoto con il proprio DBMS.Poichè l'utilizzo di un particolare DBMS esula dallo scopo di questa tesi,è stato utilizzato Microsoft Access a titolo di prova;anche se naturalmente ,volendo utilizzare l'applicazione,sarà opportuno servirsi di un DBMS più efficente.
In seguito è stata costruita una connessione ODBC in modo che l'applicazione possa interagire con il database.
Lavorando in ambiente Windows,dal Pannello di Controllo,negli strumenti di amministrazione è stata creata una nuova origine

di dati ODBC ,chiamata *Facoltàdb*.
Aprendo la vista di mapping dell'albero di progetto di WebRatio,cliccando con il tasto destro sul nodo*Data Sources* è stato selezionato il comando *Add data sources*. In questo modo viene aggiunto all'applicazione un database di supporto al quale è stato dato un nome significativo,*ProgettoWebFacolta database*.
Come mostrano le proprietà di tale database in figura 4.37 ,deve essere specificato il driver JDBC da utilizzare(JavaSoft JDBC-ODBC,in ambiente Windows) , la URL dell'origine di dati rispettando la sintassi del driver JDBC scelto(jdbc:odbc:Facoltàdb) ed eventualmente un usrename e password(non specificati in questo esempio).

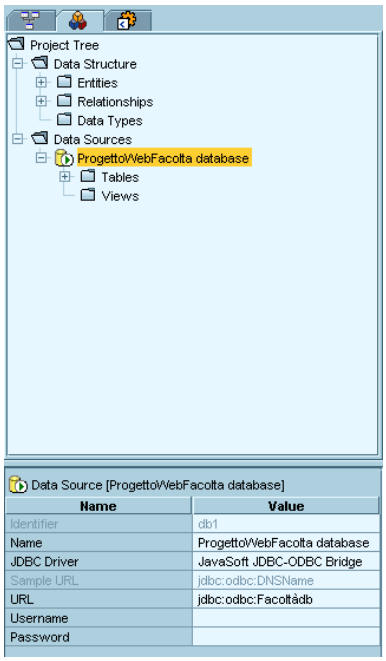

Figure 4.37: Vista di mapping dell'albero di progetto con database di supporto dell'applicazione

In seguito ,per connettere l'origine di dati e renderla accessibile,è stato selezionato il comando *Refresh* cliccando con il tasto destro sull'icona del database:WebRatio segnala se la connessione

è stata eseguita con successo e mostra la lista delle tabelle e delle viste.
Per creare il database con dati di prova è stato selezionato il comando *Create Filled Data Mapping*.WebRatio mantiene i dati di prova in file di testo in un sottodirettorio ;vi è un direttorio per ogni tipo di attributo(string,integer,..) contenente files ASCII con diversi valori.

Terminato il processo di mapping,è stato effettuato il controllo degli errori selezionando il bottone *Mapping Warnings*(figura 4.38) dalla barra principale degli strumenti.

Figure 4.38: Bottone per i warning di mapping

Specifica della Presentazione:
La presentazione viene specificata in WebRatio associando le site views,le aree,le pagine,le units,gli attributi,i campi delle forms ed i links con regole di presentazione scritte in XSL.
Tali regole non sono direttamente visibili dagli utenti di WebRatio ,poichè sono incapsulate nel concetto di *foglio di stile*.
Internamente,un foglio di stile consiste di un insieme di regole XSL che convertono le specifiche WebML in templates di apgina in un certo linguaggio di markup(HTML o WML,ad esempio).
Come già sottolineato ,WebRatio permette di definire nuovi fogli di stile mediante il tool *EasyStyler*;ciò comporta la definizione di layouts di pagine,di units e di attributi/links/fields e di look and feel.
Tuttavia WebRatio mette a disposizione anche alcuni fogli di stile predefiniti; nell'applicazione della Facoltà è stato utilizzato uno di tali fogli di stile in quanto una definizione dettagliata della presentazione esula dagli scopi di questa tesi.

Poichè ,a differenza delle specifiche della struttura e dell'ipertesto,la specifica della presentazione è dipendente dalla piattaforma,è stato selezionato *Project Tree* dalla vista di presentazione dell'albero di progetto e settata la piattaforma di sviluppo (figura 4.39).La proprietà *Layout Platform* indica la piattaforma di presentazione server-side utilizzata per lo sviluppo dell'applicazione,in questo caso JSP 1.1.

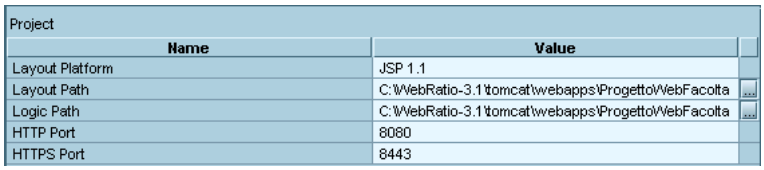

Figure 4.39: Setting della piattaforma nella specifica della presentazione

In seguito ,cliccando sull'icona di ogni site view ed utilizzando l'area delle proprietà,è stato selezionato il foglio di stile ed il layout di pagina da utilizzare per la generazione delle pagine(la figura 4.40 mostra la scelta per la site view FacoltaWeb).
Nell'applicazione sviluppata si è scelto di non specificare fogli di stile differenti a livello di aree o pagine,in modo che in modo che il foglio di stile scelto per la site view venga applicato a tutte le pagine.

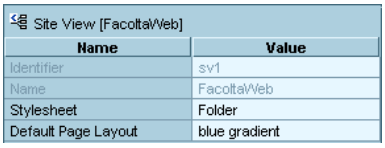

Figure 4.40: Scelta del foglio di stile e del layout di pagina per la site view FacoltaWeb

La vista di presentazione dell'albero di progetto mostra ,per ogni site view,le pagine che la compongono;selezionando l'icona di una pagina nell'area di lavoro appare una griglia.Tale griglia è stata utilizzata per selezionare le relative posizioni delle units

contenute nella pagina.
La figura figura 4.41 mostra il posizionamento delle tre units contenute nella pagina *Appelli Editing* dell'area *APPELLI* della site view *FacoltaContentManagement2*.

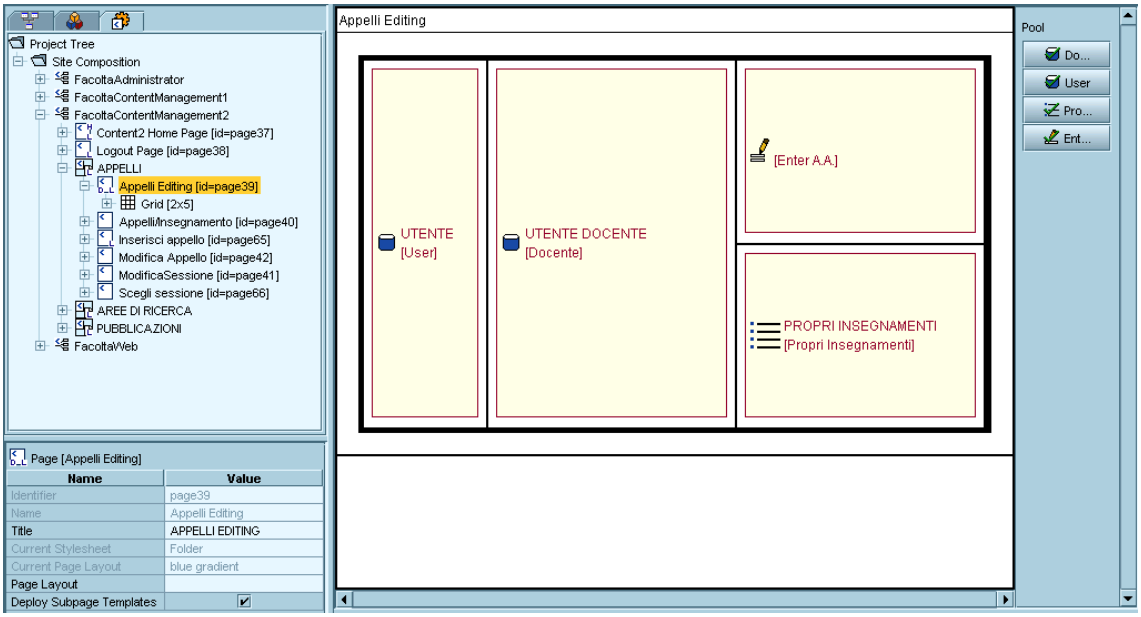

Figure 4.41: Specifica del posizionamento delle units di una pagina della site view FacoltaContentManagement2

Terminata la specifica della presentazione,è stato effettuato il controllo degli errori selezionando il bottone *Publishing Warnings*(figura 4.42) dalla barra principale degli strumenti.

Figure 4.42: Bottone per i warning di presentazione

Generazione del codice:
I templates di pagina e i descrittori XML dell'applicazione riguardante la facoltà sono stati prodotti automaticamente da WebRatio ,come descritto precedentemente.In particolare sono stati utilizzati comandi separati per produrre le due parti differenti del codice dell'applicazione.
Per ogni site view,una volta selezionata dalla vista di presentazione,sono stati generati i descrittori XML,selezionando il comando *Build XML Descriptors*(figura 4.43), ed in seguito i templates di pagina,mediante il comando *Build Page Templates*(figura 4.44).
In alternativa è possibile utilizzare il comando *Build All*(figura 4.45)per generare in una sola volta il codice completo dell'applicazione.

Figure 4.43: Bottone per la generazione dei descrittori XML

Figure 4.44: Bottone per la generazione dei templates di pagina

Figure 4.45: Bottone per i warning di presentazione

La figura 4.46 mostra la home page della site view FacoltaWeb,mentre la figura 4.47 mostra la pagina Appelli Editing dell'area APPELLI della site view FacoltaContentManagement2 della quale è stata mostrata anche la specifica della presentazione(figura 4.41).

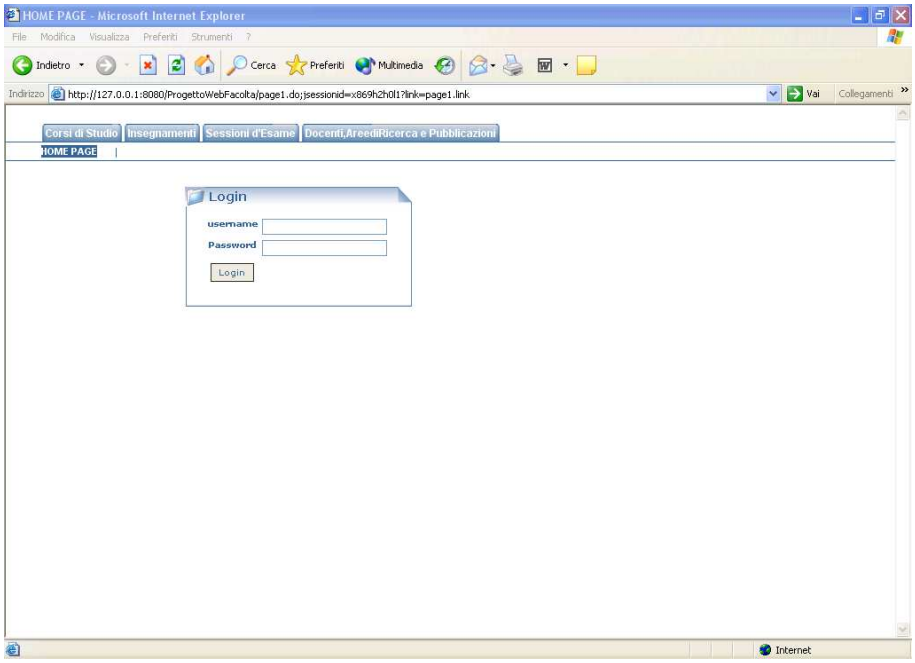

Figure 4.46: La home page della site view FacoltaWeb

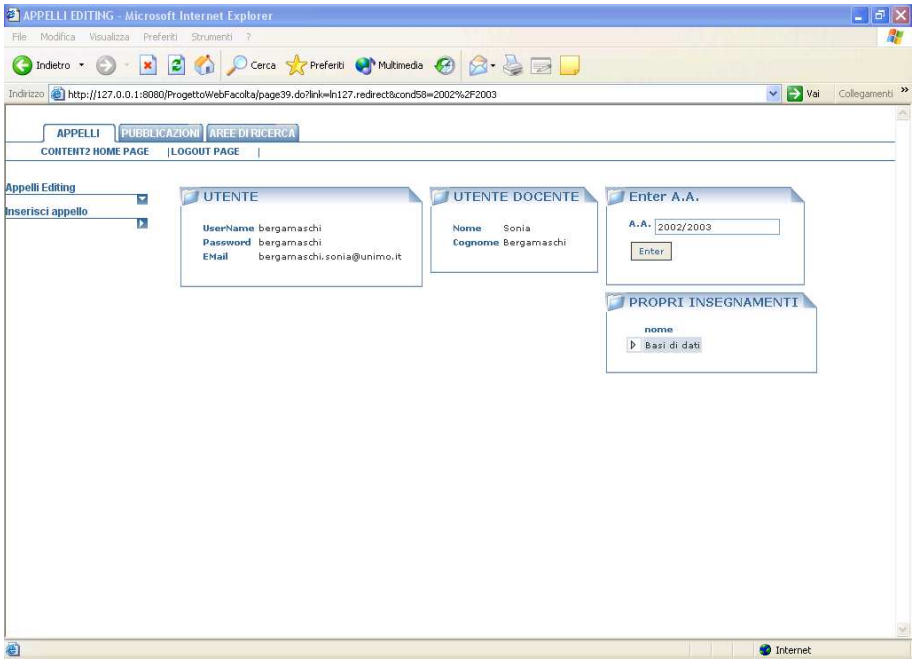

Figure 4.47: La pagina Appelli Editing dell'area APPELLI della site view FacoltaManagement2

Generazione della documentazione:
Dopo aver completato lo sviluppo dell'applicazione riguardante la facoltà ,è stata generata una documentazione formale del progetto.
WebRatio contiene un elemento per la generazione automatica della documentazione del progetto,in un formato chiamato *WebMLDoc*,basato sul popolare formato di documentazione *JavaDoc*.
Per invocare il generatore WebMLDoc,è stato selezionato dalla barra principale degli strumenti il comando *Generate WebMLDoc*,la cui icona è mostrata in figura 4.48.

Figure 4.48: Icona del comando *Generate WebMLDoc* nella barra principale degli strumenti

Prima di generare la documentazione ,WebRatio chiede all'utente di inserire una directory di output dove verranno salvati i files HTML(figura 4.49).

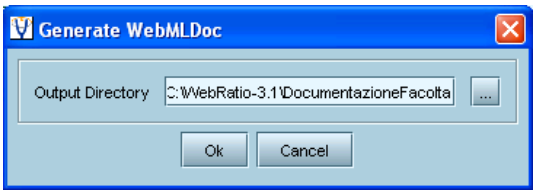

Figure 4.49: Scelta della directory di output per la documentazione di progetto

Completata la generazione,aprendo il file index.html nella directory di output,appare la home page del progetto WebMLDoc relativo all'applicazione sviluppata,da cui è possibile visualizzare tutti gli elementi del progetto(figura 4.50).

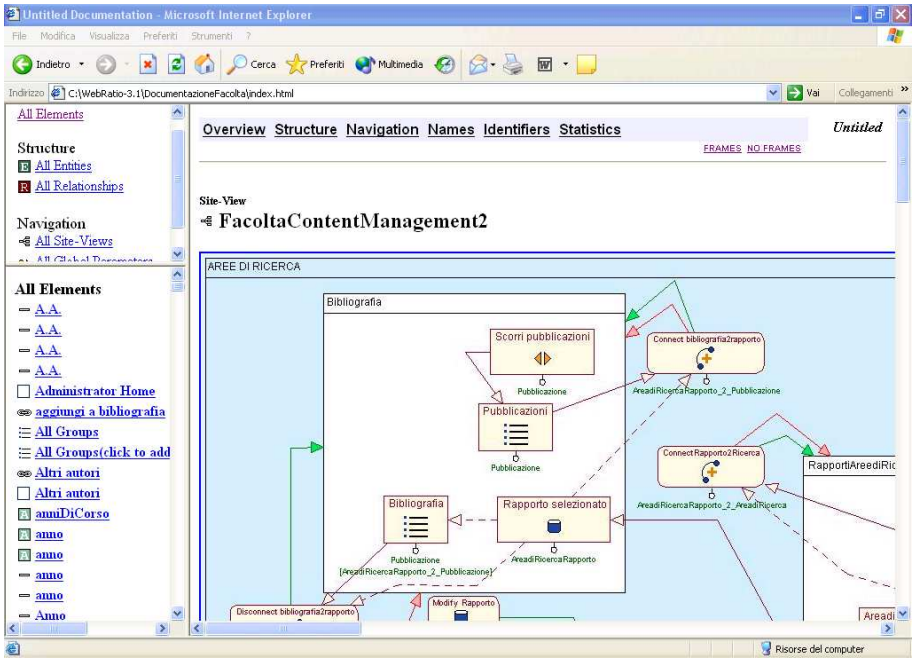

Figure 4.50: Documentazione del progetto ProgettoWebFacolta

4.1.4 Rational Rose XDE

Rational Rose XDE, o *Rational Rose E**x**tended **D**evelopment **E**nviroment*,è l'ultima edizione di *Rational Rose* ,il tool di design e analisi OO più popolare.
L'edizione di *Rational XDE* analizzata ed utilizzata è quella basata sulla piattaforma Java,ma è disponibile anche la versione per la piattaforma .NET.
Rational XDE fornisce un ricco insieme di modelli allo scopo di supportare il maggior numero di fasi del ciclo di vita del software da produrre:dalla modellazione dei dati,alla modellazione Web basata su Java,J2EE e JSP e alle capacità di round-trip engineering.
L'edizione XDE mette a disposizione componenti aggiuntivi concepiti espressamente per supportare lo sviluppo Web ed un insieme di strumenti per l'intero processo di sviluppo e di implementazione di applicazioni Web.

L'ambiente e le funzionalità di Rational XDE

L'ambiente di Rational XDE è composto da prospettive,finestre di documenti,viste,editors e menus.
La finestra di un'applicazione Rational XDE contiene una o più prospettive.
Una *prospettiva* è una particolare disposizione della finestra di un documento,oltre che un insieme di viste (o strumenti) per lavorare con i modelli o con altre risorse.
Attualmente Rational XDE supporta una singola *prospettiva di Modelling*,una *prospettiva di Help* e più *prospettiva di sviluppo Java*.
La *prospettiva di Modelling* predefinita,contiene le seguenti viste o strumenti:

- *Navigator*.permette di navigare attraverso i propri progetti.

- *Model Explorer*:permette di navigare la struttura logica di ogni modello e di aggiungere direttamente degli elementi.

- *Finestra delle proprietà*: utilizzata per mostrare e modificare le proprietà degli elementi dei modelli.

- *Barra degli strumenti*: fornisce le icone per gli elementi che possono essere utilizzati nella creazione dei modelli,suddivise in base all'ambito del software o al tipo di diagramma(oltre a tutti i diagrammi UML,sono presenti elementi per la modellazione Java e Web). *Vista Tasks*: contiene messaggi per la validazione dei modelli. *Vista Output*: visualizza messaggi provenienti da altri strumenti.

La modellazione e lo sviluppo vengono realizzati all'interno del contesto di un *progetto*.
Un progetto consiste di tutti i file collocati in una particolare directory e può contenere modelli,codice sorgente o files di testo.Vi sono diversi tipi di progetti;i tre tipi più importanti sono:

- *Progetti Basic Modelling*:per attività generali di modellazione che non comportano costrutti specifici di Java.

- *Progetti Java Modelling*:per lo sviluppo di applicazioni Java.

- *Progetti Web Modelling*:per lo sviluppo di applicazioni Web.

Rational XDE estende allo sviluppo Web i benefici della modellazione OO:

- *Reverse Engineering*:Rational XDE permette di effettuare il Reverse Engineering a partire dal codice sorgente per ottenere i modelli corrispondenti.

- *Generazione del codice e sincronizzazione manuale/automatica*:Rational XDE fornisce elevate capacità di sviluppo di codice Java;infatti

include ununun editor di codice Java,un compilatore,uno strumento di build ed un debugger locale/remoto.In particolare è possibile realizzare il round-trip engineering(RTE) del proprio codice Java.
Modelli speciali,detti *Code Models*,vengono utilizzati per descrivere il round-trip engineering dell'attuale implementazione Java della propria applicazione.
Ogni progetto può contenere un solo *code model* che può essere associato a tutte o ad alcune delle sorgenti Java del progetto.
Una volta creata un'associazione tra il codice del progetto ed il suo *code model*,il modello ed il codice possono essere sincronizzati utilizzando tecniche di sincronizzazione automatica o selettiva.

- *Patterns e Templates di codice*:in Rational XDE un pattern è implementato come ina collezione di elementi,relazioni e comportamenti UML,ed è costruito in modo da accettare argomenti in ingresso.L'utente può dunque ,quando utilizza un pattern,inserire determinati valoriper tali argomenti sulla base del particolare contesto di modellazione nel quale verrano utilizzati gli elementi creati dall'espansione del pattern.
  I templates di codice permettono invece di automatizzare generazioni ripetitive di codice.

- *Sviluppo multi-modello*:Rational XDE permette di utilizzare modelli e diagrammi di forma libera,per avantaggiare diagrammi di notazioni definite dall'utente;inoltre l'utente può decomporre e mantenere il proprio modello ad ogni livello di astrazione.

- *Modellazione del database*:Rational XDE permette di modellare database e di integrare il design delle applicazioni con

quello dei dati.

- *Pubblicazione sul Web e creazione di Reports*: è possibile generare pagine Web basate su HTML che contengono viste "read-only" dei propri modelli .Rational XDE presenta la capacità,basata sui templates,di creare reports basati su HTML per documentare i contenuti dei modelli.Tali reports sono adatti per essere stampati,copiati e distribuiti.

La modellazione Web con Rational XDE:un esempio pratico

Nell'ambito della modellazione web,Rational XDE si basa sull'estensione di UML per modellare le applicazioni Web proposta da Conallen (paragrafo 3.3.3).
Rational XDE adotta,oltre ai meccanismi tradizionali di estensione UML come gli stereotipi,i vincoli ed i valori etichettati,la considerazione di base dell'approccio di Conallen:la divisione degli artefatti web in oggetti lato-server ed oggetti lato-client.
In base a tale suddivisione vengono modellate pagine server per gli oggetti che interagiscono e dipendono da risorse lato-server e pagine client per gli oggetti eseguiti sulla macchina del client utilizzando risorse del browser.

In Rational XDE un'applicazione Web viene costruita utilizzando tre modelli che interagiscono tra loro:

- *Modello Directory Virtuale*:contiene classi e relazioni stereotipate rappresentanti JavaServer Pages,documenti e forms HTML e le relazioni tra questi artefatti Web.

- *Modello JSP Tag Lib*:contiene classi e relazioni che rappresentano rispettivamente le abituali etichette JSP e le relazioni di dipendenza per etichettare le classi di un modello java o le pagine JSP in un modello directory virtuale.

- *Modello Java Code*: contiene classi JavaBeans che sono in relazione con le pagine JSP nel modello directory virtuale ed altri costrutti Java utilizzati nelle applicazioni Web,come EJBs e servlets.

Rational XDE offre la possibilità di modellare un'applicazione Web basandosi su due possibili approcci:

- Effettuando il *reverse engineering* dei files sorgenti di un'applicazione Web esistente per creare il modello dell'applicazione web.
- Aggiungendo al modello directory virtuale delle classi stereotipate rappresentanti artefatti web e definendo relazioni tra le classi.

Nell'analisi delle varie fasi per la modellazione di applicazioni web e nello sviluppo di un'esempio pratico ,si è seguito il secondo tipo di approccio,cioè si è supposto di dover sviluppare l'applicazione per la prima volta.

L'esempio sviluppato con Rational XDE modella la pubblicazione delle informazioni riguardanti le Sessioni d'esami con i relativi appelli ed esami di laurea ,modellata in WebRatio Site Development Studio all'interno dell'area *Sessioni d'Esame* della site view FacoltaWeb .
In questo modo sarà possibile,limitatamente a tale esempio,confrontare i due differenti approcci nella modellazione di applicazioni Web.

Nello sviluppo dell'esempio descritto sono state eseguite le principali operazioni per la modellazione e lo sviluppo di applicazioni Web mediante Rational XDE.

1. Innanzitutto è necessario creare un nuovo progetto Web,selezionando dal menu *File*l'opzione *New*,seguita dall'opzione *Project* e dalla scelta *Web Modelling Project*come tipo di progetto.

Nel *Navigator* viene creato un folder corrispondente al progetto ed i tre modelli(Directory Virtuale,JSP Tag Library e Java Code)vuoti,tutti con l'estensione *.mdx*.Rational XDE utilizza il nome del progetto per nominare i modelli,tuttavia c'è la possibilità di dare ai modelli dei nomi che permettano di identificarli più facilmente.
La figura 4.51 mostra la vista *Navigator* per il progetto d'esempio,chiamato *EsempioSessioni*,contenente i tre diagrammi.

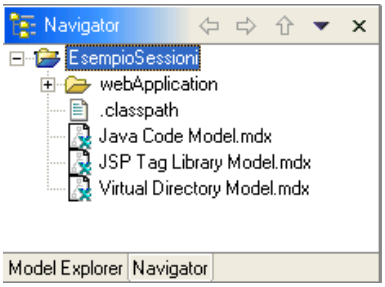

Figure 4.51: Vista *Navigator* del progetto *EsempioSessioni*

2. Una volta creato il progetto si possono iniziare a costruire i modelli,in particolare il modello Directory Virtuale.
   La directory virtuale è un package stereotipato contenente classi stereotipate che rappresentano artefatti Web ed associazioni stereotipate tra esse.

   Basandosi sulla separazione tra lato server e lato client proposta da Conallen,Rational XDE permette di inserire *pagine client* e *pagine server*.
   Viene utilizzata una classe stereotipata *ServerPage* per modellare le funzionalità lato server,in particolare per creare una *JavaServer Page (JSP)*.
   È possibile inserire una pagina server nel modello selezionando la relativa icona nella barra degli strumenti all'interno

della categoria Web, mostrata in figura,o cliccando con il tasto destro sulla directory virtuale nella vista *Model Explorer*,selezionando l'opzione *Add Java* e quindi *Server Page*,come mostra la figura.
Quando si inserisce nel modello una *Server Page*,XDE automaticamente aggiunge una *Client Page* e crea una relazione stereotipata *Build* tra le due classi per indicare che la pagina client viene costruita dalla pagina server.

3. Rational XDE permette di inserire anche pagine client non costruite da una JSP,dette *standalone*,per mezzo di una classe stereotipata *Client Page*,il cui inserimento è analogo a quello di una pagina server.

4. È possibile inserire nel modello anche il principale meccanismo di data-entry delle applicazioni Web: le *Forms HTML*.Per modellare l'interazione con l'utente viene aggiunta una classe stereotipata *HTMLForm* ad una pagina client.Tale classe stereotipata non può contenere operazioni ed i suoi attributi rappresentano campi di input HTML.Tali campi possono essere di diversi tipi:submits,text,multiple line text o radio button.
Come indicato, una form HTML viene aggiunta ad una pagina client e l'associazione tra tali classi viene rappresentata automaticamente da XDE come un'associazione di aggregazione,in quanto la form è completamente contenuta nel documento HTML rappresentato dalla pagina client.
Ogni form HTML deve inoltre essere posta in relazione con la pagina server che ne processa i dati di input;ciò avviene mediante una relazione stereotipata *Submit*.

5. Per modellare il passaggio di controllo da una JSP ad un'altra pagina server o ad una pagina client viene utilizzata la relazione stereotipata *JSPForward*.
   Per modellare l'utilizzo da parte di una JSP del contenuto di un'altra pagina server o di una pagina client si utilizza la relazione stereotipata *JSPInclude*.

6. Rational XDE permette di modellare anche il passaggio ad una JSP di dati memorizzati in un JavaBean.
   Per effettuare tale modellazione è necessario,nella vista model Explorer, cliccare con il tasto destro sulla pagina server corrispondente ,selezionare l'opzione *Add Java* e quindi *New JavaBean*.Rational XDE aggiunge una nuova classe Java nel modello Java Code ed una relazione stereotipata *JSPUseBean* tra la pagina server e la classe Java rappresentante il Bean nel modello directory virtuale.

7. Un'altro tipo di relazione stereotipata predefinita in XDE è la relazione *HTMLLink*,utilizzata per modellare un cammino navigazionale tra due pagine client o tra una pagina client ed una pagina server.Tale relazione ha origine da una pagina client,in quanto corrisponde ad un link ipertestuale tra pagine.

   Nell'esempio sviluppato sono stati inserite alcune delle classi e relazioni stereotipate descritte.
   In figura 4.52 è mostrata la parte del modello directory virtuale del progetto EsempioSessioni nella quale vengono mostrate le sessioni relative ad un anno accademico inserito dall'utente.
   Già in questa parte sono state utilizzate una form HTML per l'inserimento dell'anno accademico,le pagine server per

recuperare le sessioni d'esame ,la pagina client per visualizzarle e la classe JavaBean SessioneEsame per contenere i dati relativi alle sessioni d'esame.

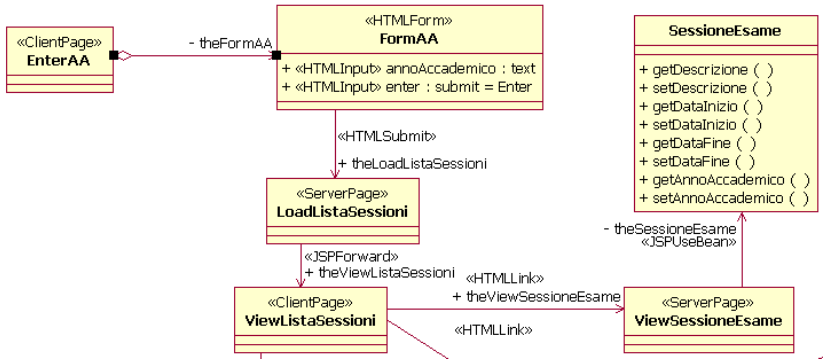

Figure 4.52: Porzione del modello directory virtuale del progetto EsempioSessioni

La figura 4.53 mostra la form HTML *FormAA* all'interno della pagina client *EnterAA* nella vista Model Explorer con i propri elementi di input.

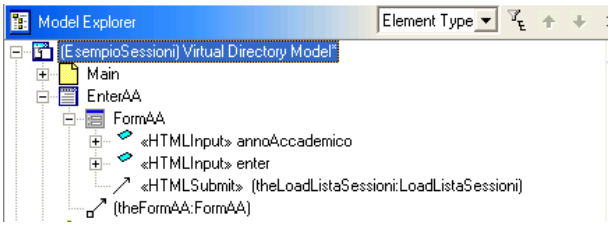

Figure 4.53: La form FormAA nella vista Model Explorer

La figura 4.54 mostra,sempre all'interno del Model Explorer,la pagina server *ViewSessioneEsame* contente la relazione *JSPUseBean*.

Utilizzando le stesse classi e relazioni stereotipate sono stati modellati anche gli appelli e gli esami di laurea con le informazioni relative,rispettivamente in figura 4.55 e in figura 4.56.

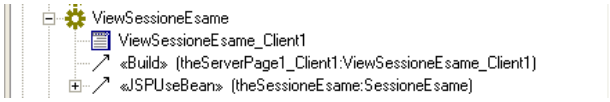

Figure 4.54: La relazione JSPUseBeans nella pagina ViewSessioneEsame nella vista Model Explorer

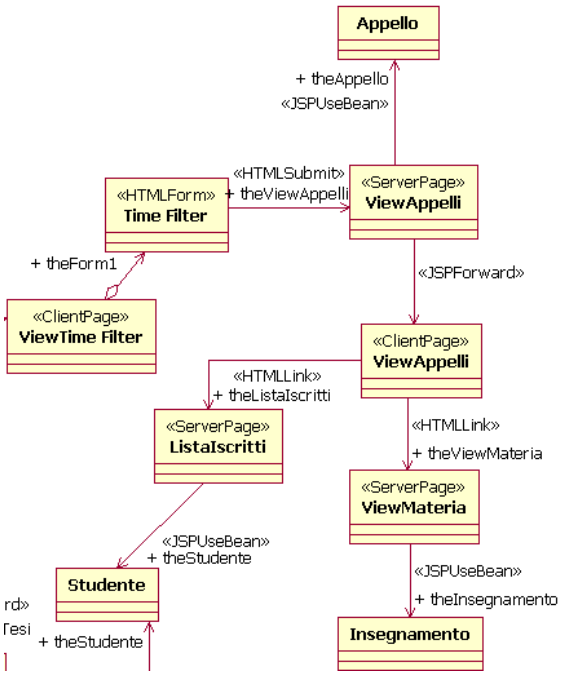

Figure 4.55: Porzione del modello directory virtuale del progetto EsempioSessioni

8. Dopo aver creato i modelli ,Rational XDE permette di generare il codice.
   È possibile generare il codice dell'applicazione durante la creazione del modello per testare individualmente gli artefatti Web inseriti e dunque come ulteriore validazione del modello stesso;oppure si può decidere di aspettare a generare il codice finchè il modello non è completo.
   XDE utilizza gli stereotipi delle classi per determinare il tipo di file da generare.

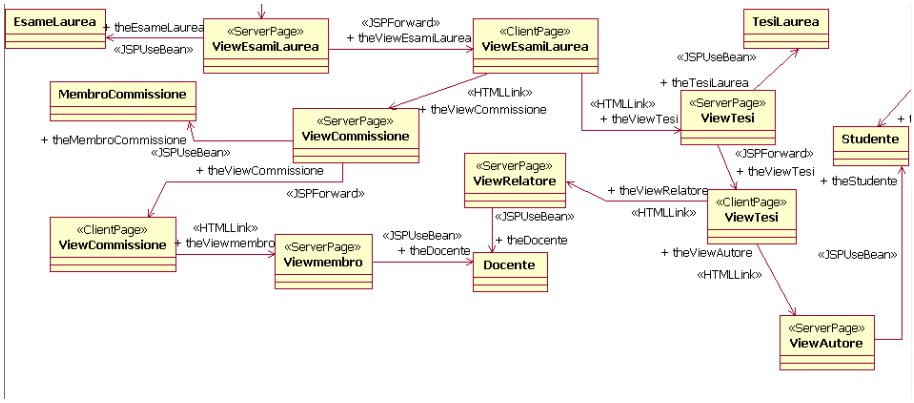

Figure 4.56: Porzione del modello directory virtuale del progetto EsempioSessioni

La classe stereotipata *Server Page* genera file *.jsp*.Quando una pagina server include una classe stereotipata *Client Page*,i tags HTML della pagina client vengono incorporati nel file *.jsp*.
Una pagina client che non è incorporata in una pagina server genera un file *.html* o *.htm*.
Durante la modellazione dell'applicazione Web è possibile settare,nella finestra delle proprietà ,le *opzioni di round-trip engineering(RTE Options)* per la pagine server e le pagine client.
Tra tali opzioni vi sono la proprietà di *File name*,in cui si può specificare il nome e l'estensione del file corrispondente ad una certa pagina,e la proprietà di *RTE Syncronization* che permette di sincronizzare la generazione di codice per artefatti Web specifici.In particolare si utilizza un attributo di sincronizzazione dal valore booleano :quando tale attributo è settato a VERO Rational XDE genera il codice,quando è settato a FALSO Rational XDE non genera codice per l'artefatto Web.

4.1.5 Confronto tra i due strumenti utilizzati

Dalla descrizione degli strumenti *WebRatio Site Development Studio* e *Rational Rose XDE* e dalle applicazioni d'esempio sviluppate con essi sono evidenti due approcci molto differenti nella modellazione e sviluppo delle applicazioni Web,ogniuno con i propri vantaggi e le proprie limitazioni.
Le caratteristiche ,positive o negative,di questi due CASE tools rispecchiano quelle evidenziate nella descrizione ed analisi dei linguaggi di modellazione che essi adottano,WebML(vedi paragrafo 3.1.3) per WebRatio e UML esteso di Conallen(vedi paragrafo 3.3.3) per Rational XDE,che probabilmente rappresentano i due esempi più significativi dei principali approcci alla modellazione Web.
WebRatio Site Development Studio modella il front-end ipertestuale delle applicazioni Web(vedi paragrafo 1.3.2) ad un alto livello di astrazione in termini di pubblicazione dell'informazione,suddivisione in pagine ed aree contenenti tale informazione e navigazione.Il linguaggio di modellazione adottato,WebML,non tiene conto della distribuzione delle funzionalità tra i componenti dell'architettura di un'applicazione Web:ciò permette di poter specificare dettagliatamente aspetti navigazionali e di presentazione,ma non di modellare i meccanismi tecnologici ed i passaggi di controllo tra client e server che concretamente realizzano determinate operazioni.
Al contrario *Rational Rose XDE* modella le applicazioni Web distinguendo le risorse lato client da quelle lato server e quindi permette di rappresentare due aspetti funzionali differenti delle pagine di un'applicazione Web :le pagina client e le pagine server. *Rational Rose XDE* adotta le estensioni di UML proposte per la modellazione Web e presenta quindi limitazione analoghe a tali estensioni:utilizzando una tecnica di modellazione "generale" come UML,non vengono fornite notazioni adeguate per

modellare navigazione e presentazione ,ad esempio.
Dal punto di vista della generazione del codice,WebRatio genera automaticamente tutto il codice necessario per l'esecuzione dell'applicazione Web,mentre Rational XDE presenta una capacità di generazione del codice piuttosto limitata,infatti dai profili UML *Rational XDE* è in grado di derivare template di pagina iniziali che lo sviluppatore può completare a mano inserendo il codice che rappresenta la logica applicativa,come i metodi delle classi,ad esempio.

Bibliography

[1] P.Chen. The entity-relationship model-toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, mar 1976.

[2] J.Rumbaugh, J.Jacobson, and G.Booch, editors. *The Unified Modelling Language Reference Manual*. Addison-Wesley, 1998.

[3] A.Gu, B.Henderson-Sellers, and D.Lowe. Web modelling languages:the gap between requirements and current exemplars. 2002.

[4] W.Retschitzegger and W.Schwinger. Towards modelling of dataweb applications-a requirement's perspective. 2000.

[5] F.Halasz and M.Schwartz. The dexter hypertext reference model. *CACM*, 37(2):30–39, feb 1994.

[6] F.Garzotto, P.Paolini, and D.Schwabe. Hdm:a model-based approach to hypertext application design. *ACM Transactions of Information Systems*, 11(1):1–26, jan 1993.

[7] F.Garzotto, L.Mainetti, and P.Paolini. Designing modal hypermedia applications. In *In proceeding of the Eigth ACM Conference on Hypertext*, University of Southampton,UK, apr 1997.

[8] P.Fraternali and P.Paolini. Model-driven development of web applications:the autoweb system. *ACM Transactions on Office Information Systems*, 18(4), 2000.

[9] S.Ceri, P.Fraternali, and A.Bongio. Web modelling language(webml):a modelling language for designing web sites. In *In proceeding of WWW9/Computer Networks33*, pages 137–157.

[10] S.Ceri, P.Fraternali, A.Bongio, M.Brambilla, S.Comai, and M.Matera, editors. *Designing Data-Intensive Web Applications*. Thr Morgan-Kaufmann Series in Data Management Systems,Jim Gray-Series Editor, dec 2002.

[11] T.Isakowitz, E.A.Sthor, and P.Balasubramanian. Rmm:a methodology for web publishing. *CACM*, 38(8):34–44, aug 1995.

[12] P.Atzeni, G.Mecca, and P.Merialdo. Design and implementation of data-intensive web sites. In *In proceeding of the Conference On Extended Database Technology(EDBT'98)*, Valencia,Spain, 1998.

[13] F.M.Fernandez, D.Florescu, J.Kang, A.Levy, and D.Suciu. Catching the boat with strudel:experiences with a web-site management system. In *In proceeding of the ACM SIGMOD International Conference on Management of Data*, pages 414–425, 1998.

[14] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, and W.Lorenson, editors. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

[15] G.Rossi, D.Schwabe, and D.J.Barbosa. Systematic hypermedia application design with oohdm. In *In proceeding of the 7th ACM Conference on Hypertext'96*, page 166, 1996.

[16] G.Rossi, D.Schwabe, and F.Lyardet. Web applications models are more than conceptual models. In *In proceeding of the World Wide Web and Conceptual Modelling'99 Workshop,ER'00 Conference*, pages 239–253. Springer.

[17] G.Rossi, D.Schwabe, and F.Lyardet. Abstraction and reuse mechanisms in web applications models. In *In proceeding of the World Wide Web and Conceptual Modelling'00 Workshop,ER'00 Conference*, pages 76–88. Springer.

[18] J.Gomez, C.Cachero, and O.Pastor. Extending a conceptual modelling approach to web application design. In *In proceeding of the 12th Conf.Advanced Information Systems(CAISE 00)*, pages 79–93, Berlin, jun 2000. Springer-Verlag.

[19] J.Conallen. Modeling web applications architectures with uml. *Communications of the ACM*, 42(10), oct 1999.

[20] J.Conallen. *Building Web Applications with UML-2th edition*. Addison-Wesley, oct 2002.

[21] Using uml to design hypermedia applications. Technical report, Institut fur informatik, Ludwig-Maximillians-University,Munchen,Germany, mar 1999.

[22] H.Baumeister, N.Koch, and L.Mandel. Towards a uml extension for hypermedia design. In R.France and B.Rumpe editors, editor, *In proceeding of the Unified Modeling Language Conference:Beyond the Standard(UML'1999)*, pages 614–629. Springer Verlag.

[23] N.Koch, H.Baumeister, R.Hennicker, and L.Mandel. Extending uml to model navigation and presentation in web applications. In R.France and B.Rumpe editors, editor, *In proceeding of the Modelling Web Applications in the UML Workshop-UML'2000*, York,England, oct 2000.

[24] L.Baresi, F.Garzotto, and P.Paolini. From web sites to web applications:new issues for conceptual modelling. In *In proceeding of ER'00*, pages 89–100. Springer.

[25] L.Baresi, F.Garzotto, and P.Paolini. Extending uml for modelling web applications. In *In proceeding of the 34th International Conference on System Sciences.*

[26] W.Kim. Advanced database systems. *ACM Press*, 1994.

[27] O.Pastor et al. From object-oriented conceptual modelling to automated programming in java. In *In proceeding of the International Conference on Entity Relationship Approach (ER98)*, pages 183–196, Berlin, 1998. Springer Verlag.

[28] A.Kraus and N.Koch. Generation of web applications from uml models using an xml publishing framework. In *In proceeding of the Integrated Design and Process Technology Conference*, Pasadena, 2001.