INDICE

Prefazione

Lo scenario attuale del mondo del web vede uno sviluppo velocissimo e caotico. Infatti aumenta esponenzialmente il numero delle persone che hanno accesso ad Internet, creando così una richiesta massiccia di applicazioni web, da sviluppare sempre in tempi brevissimi. Questa particolare situazione fa sì che la disciplina del Web Engineering sia ancora ai suoi albori e non esistano metodologie e modelli descrittivi universalmente riconosciuti. Siccome tale situazione si protrae già da qualche anno, si sta presentando anche l'esigenza di analizzare applicazioni già esistenti, avendo come obiettivo l'analisi, la manutenzione oppure il reverse engineering.
La tesi si occupa del Reverse Engineering di un'appliczione web, cercando una metodologia generale e specializzandosi sulla realizzazione automatica di una sua fase.

Nel primo capitolo viene introdotta la discplina del web engineering, con cenni alle principali problematiche relative alle applicazioni web. In particolare vengono descritte alcune delle proposte di ciclo di vita presenti in letteratura per tali appicazioni.
Il secondo capitolo focalizza l'attenzione sui modelli architetturali più diffusi sui quali le applicazioni web si basano.
Il terzo capitolo riporta una panoramica sulle più diffuse tecnologie a supporto delle applicazioni web, effettuandone anche un raffronto critico.
Il quarto capitolo é dedicato alla definizione di una serie di modelli e metodi che permettano di progettare efficacemente un'applicazione web. A tale scopo si é adattato lo UML, estendendone il campo di applicazione e la semantica dei suoi elementi e dei suoi diagrammi. E' stata inoltre applicata questo modello di rappresentazione nel progetto, in forward engineering, di una semplice applicazione web di esempio.
Nel quinto capitolo viene introdotto il problema generale del reverse engineering, cercando poi una sua specializzazione al caso delle applicazioni web. Al fine di trarre una metodologia, é stato analizzato in reverse engineering una semplice applicazione web.
Nel sesto capitolo é stata formulata una metodologia di reverse engineering per applicazioni web, ricavate sulla base dell’esperienza fatta con l'esempio analizzato.

Di questa metodologia é stata definita una sua suddivisione in fasi e sono stati definiti esplicitamente tutti gli input e gli output.

Con il settimo capitolo inizia la parte applicativa del lavoro di tesi. Viene definito l'obiettivo: realizzare uno strumento automatico che sia di supporto alla prima fase del processo di reverse engineering, riguardante l'analisi statica del codice sorgente di un'applicazione web. Sono, quindi, definiti i requisiti del tool che si vuole sviluppare, la sua architettura e il suo campo di applicazione. Viene fatta un'ulteriore suddivisione del progetto, in modo da individuare tutte le fasi e gli elementi necessari alla realizzazione di un estrattore di codice sorgente, che possa fungere da punto di partenza per l'analizzatore statico.

Il capitolo ottavo si occupa dell'analisi delle informazioni da estrarre dal codice sorgente, concretizzando lo studio di queste ultime nella definizione del modello di un database atto a contenerle e gestirle.

Il capitolo nono si occupa invece della traduzione delle informazioni, dal codice sorgente verso una rappresentazione intermedia che possa essere di input ad un tool astrattore per popolare il database con le informazioni raccolte, nonchè realizzare astrazioni di livelli più alti. In questo capitolo, quindi, sono definite le informazioni da estrarre nel codice sorgente ed è definita la sintassi e la semantica dei file che registrano la forma di rappresentazione intermedia.

Nel capitolo decimo è descritto lo sviluppo del tool estrattore realizzato, ed é riportata la sua documentazione, e fornite le istruzioni per usarlo.

Nel capitolo undicesimo si é provveduto a validare i risultati del tool realizzato, nonchè della metodologia ipotizzata, effettuando il reverse engineering di due casi di studio.

Nel capitolo dodicesimo sono state proposte alcune possibili estensioni future del tool, che mirano alla copertura di tutto il processo di reverse engineering di un’applicazione web e alla sua integrazione con strumenti di forward engineering.

Nel capitolo tredicesimo sono riportate le conclusioni che si sono tratte da tutto il lavoro di tesi.

CAPITOLO 1 - INTRODUZIONE

1 Introduzione

Con il diffondersi di Internet come mezzo di scambio di informazioni tra utenti distanti, è nata la necessità di non condividere solo informazioni in semplici pagine web accessibili da un qualsiasi browser, ma di aggiungere interattività e di realizzare applicazioni ben più complesse rivolte ad un vasto pubblico di utenti più o meno esperti. Per sviluppare tali applicazioni è stata utilizzata come piattaforma il World Wide Web, grazie alla possibilità di integrare multimedialità e funzionalità nelle stesse pagine e di rendere il tutto disponibile ad utenti di differenti piattaforme Software e Hardware.

In molti casi si confonde un'applicazione Web con un sito Web. Si può dire che un sito Web è il front-end di una applicazione Web, quindi provvede alla presentazione all'utente e all'organizzazione delle informazioni, mentre un' applicazione Web è qualcosa di più. Un utente interagisce direttamente con una applicazione Web, fornendo dati di input e attendendo un'elaborazione di questi. Tale elaborazione non è come una semplice ricerca di un'informazione, ma influenza direttamente lo stato dell'applicazione. Un esempio pratico è quello di una vendita on-line (e-commerce). In questo caso l'ordine dell'utente causa una transazione nell'archivio dei prodotti e una transazione nel conto corrente con cui è stato fatto l'acquisto. Quindi un altro aspetto è che più entità sono interessate all'applicazione, in un sito web abbiamo una semplice interazione tra l'utente e il web server.

2 Web Engineering

Le prime applicazioni Web realizzate prevedevano una metodologia di sviluppo ad hoc per il particolare problema, senza un approccio sistematico ben definito. Ancora oggi non esiste una metodologia precisa, ma negli ultimi anni sta nascendo un settore dell'ingegneria del software specificamente rivolto allo sviluppo di applicazioni sul web: Web engineering.

L'introduzione del web engineering è orientata ad impedire una crisi nello sviluppo di applicazioni sul web. Lo sviluppo di applicazioni complesse senza una solida metodologia rischia di avere un'alta probabilità di fallimento. Lo stesso fenomeno si era avuto nello sviluppo di software tradizionale dove l'introduzione delle metodologie dell'ingegneria del software ha prodotto un enorme vantaggio sia durante lo sviluppo sia per attività successive al rilascio del prodotto (manutenzione).

Nonostante il web engineering adotti la maggior parte dei principi dell'ingegneria del software si possono notare alcune peculiarità proprie del web engineering:

- La maggior parte delle applicazioni web esistenti sono progettate con lo scopo di registrare informazioni e presentarle all'utente;
- Lo sviluppo di applicazioni web ha come obiettivo principale la visualizzazione e quindi non può prescindere da un accurato progetto dell'interfaccia utente;
- La molteplicità dei profili di utente rende lo sviluppo del front-end più complicato;
- Nella maggior parte dei casi un'applicazione web richiede che sia sviluppata in un tempo molto breve, ad esempio per rispettare esigenze di mercato. Risulta quindi difficile applicare completamente e rigidamente metodi e tecniche dell'ingegneria del software ed eseguire un'attività di testing che sia esaustiva.

Il web engineering risulta in una fusione di attività provenienti da differenti discipline: software engineer, multimedialità, linguaggi ipertestuali,… In figura 1 è visualizzata l'interazione del web engineering con le altre discipline.

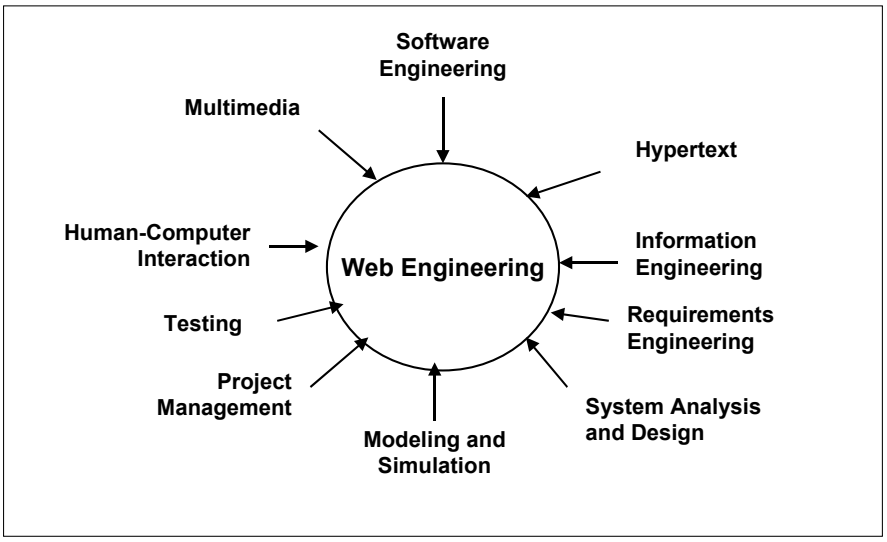


Figura 1 Discipline legate al web engineering

Il web engineering non è un'unica attività, ma è composto da numerosi task che partono dalla definizione iniziale dei requisiti fino all'implementazione. Di tali task ancora non sono stati introdotti metodi e tecniche che possano aiutare in una descrizione precisa.

3 Problematiche ed obiettivi delle applicazioni web

E' possibile redigere un elenco di quelli che possono essere i principali requisiti di una web application, ovvero di quei requisiti che divengono più pressanti in essa rispetto ad un'applicazione software tradizionale:

- privatezza, poiché l'ambito distribuito dei sistemi implica lo scambio di informazioni talvolta private. Tale requisito si accompagna quindi a tecniche di crittografia in modo da rendere criptata l'informazione scambiata. Tale requisito implica nuove procedure di testing orientate a stabilire il corretto funzionamento delle tecniche di crittografia;
- sicurezza, il sistema deve essere immune alla possibile corruzione dei messaggi, che potrebbe inficiare il corretto funzionamento, e quindi la coerenza del sistema;
- scalabilità, poiché un aspetto tipico é quello che le applicazioni web sono dirette ad un numero elevato e variabile di utenti. Devono essere perciò tenute in conto tutte le problematiche dovute agli accessi contemporanei a tutte le possibili parti del sistema, e ciò implica che anche il testing deve seguire procedure nuove rispetto a quelle tradizionali;
- interoperabilità con sistemi esistenti e portabilità. Poiché uno dei pregi di una web application é quella di poter essere utilizzabile dal maggior numero di utenti possibili, é necessario che essa possa essere raggiunta con qualsiasi piattaforma, anche "legacy". Allo stesso modo deve poter essere possibile anche elaborare dati preesistenti, in modo da poter ottenere una coesistenza con i sistemi esistenti. Un ulteriore obiettivo che si sta perseguendo oggi é quello di rendere queste applicazioni indipendenti dal linguaggio di programmazione nel quale esse sono sviluppate (ad esempio le applicazioni in ambiente CORBA);
- usabilità, in modo da allargare il più possibile il bacino di utenza potenziale dell'applicazione, astraendola ancor più dal contesto della sua implementazione. Questa considerazione può portare ad un modello di sviluppo user-centered;
- qualità visuale, che é un obiettivo un po' avulso dal campo tradizionale del software engineering, in quanto riguarda anche la gradevolezza della presentazione. Per queste ragioni si vede come il team di sviluppo di una web application debba essere più del solito variegato e costituito da persone con diversa cultura informatica.

4 Ciclo di vita di una Applicazione Web

Come detto in precedenza non esistono ancora metodologie affermate per sviluppare una Web-Application, e quindi definire un ciclo di vita non è semplice. Possiamo recuperare sicuramente nozioni che ci vengono dall'Ingegneria del Software, visto che alla fine una Web Application è un'applicazione software in un ambiente distribuito. Le proposte esistenti attualmente in letteratura tentano di adattare i classici modelli di ciclo di vita del software al caso delle Web application. In questo paragrafo vengono presentati alcuni modelli, ma soprattutto si vuole focalizzare l'attenzione su alcune attività che assumono un ruolo centrale nello sviluppo di applicazioni Web.

Un modello di ciclo di vita che è possibile applicare è quello prototipale. In una Web application infatti ha una fondamentale importanza il modo con cui l'applicazione si presenta all'utente, il che fa assumere all'interfaccia utente un ruolo di primo livello nel ciclo di sviluppo. Può essere inoltre più difficile per tali applicazione stabilire un insieme di requisiti da specificare all'inizio del ciclo di sviluppo e quindi c'è bisogno di definire qualche requisito generale da cui realizzare un prototipo da utilizzare per raffinare l'analisi iniziale.Il prototipo così realizzato, come sempre avviene anche nel campo del software tradizionale, diventa il punto di partenza del progetto.

Si vuole ora analizzare qualche modello di ciclo di vita di applicazioni web trovato in letteratura.

Il modello di seguito descritto è stato proposto da Fraternali [3].

Nella fase iniziale di *analisi dei requisiti* si vanno a identificare la tipologia di utenti, il dominio applicativo e quindi il tipo di informazioni trattate dall'applicazione. Particolare attenzione va all'interazione Uomo-macchina che si può adattare al particolare utente e deve consentire di ritrovare facilmente le informazioni necessarie. L'analisi dei requisiti è più orientata alla definizione dei profili di utente e dei requisiti legati all'interazione con l'applicazione, l'analisi concettuale è orientata invece alla ricerca di un modello il più possibile indipendente dall'implementazione. E' proprio quest'indipendenza il fattore da perseguire al fine di semplificare il più possibile le fasi di evoluzione e manutenzione, nonché la possibilità di riuso.

Una volta definiti i requisiti, l'applicazione va strutturata nei vari componenti facendo riferimento al tipo di architettura che si vuole adottare, ma senza particolari

riferimenti alle tecnologie implementative che verranno utilizzate nella fase successiva. La fase di *design* si basa su tre dimensioni principali:

- strutturale: viene descritta l'organizzazione delle informazioni gestite dalla WA mediante oggetti e relazioni tra essi.
- navigazione: sono definiti i vari percorsi all'interno dell'applicazione per accedere ad informazioni e a funzionalità
- presentazione: si concentra sul progetto dell'interfaccia utente e gli eventi attivi che causano un cambiamento di stato nell'applicazione (transazioni, inserimento in un database, ...)

Nella fase *implementativa* si deve prima specificare l'architettura e la tecnologia da utilizzare e poi si provvede all'implementazione vera e propria che non comprende solo il linguaggio di programmazione, ma anche come implementare le interazioni tra i vari elementi che costituiscono l'architettura generale (in seguito verranno chiariti i dettagli delle molteplici tecnologie realizzative).

Nel modello di Fraternali si può notare come non siano state enfatizzate alcune attività cruciali come il rilascio dell'applicazione software e il testing che invece è stato inglobato nella fase di manutenzione. Il rilascio prevede una complessa attività

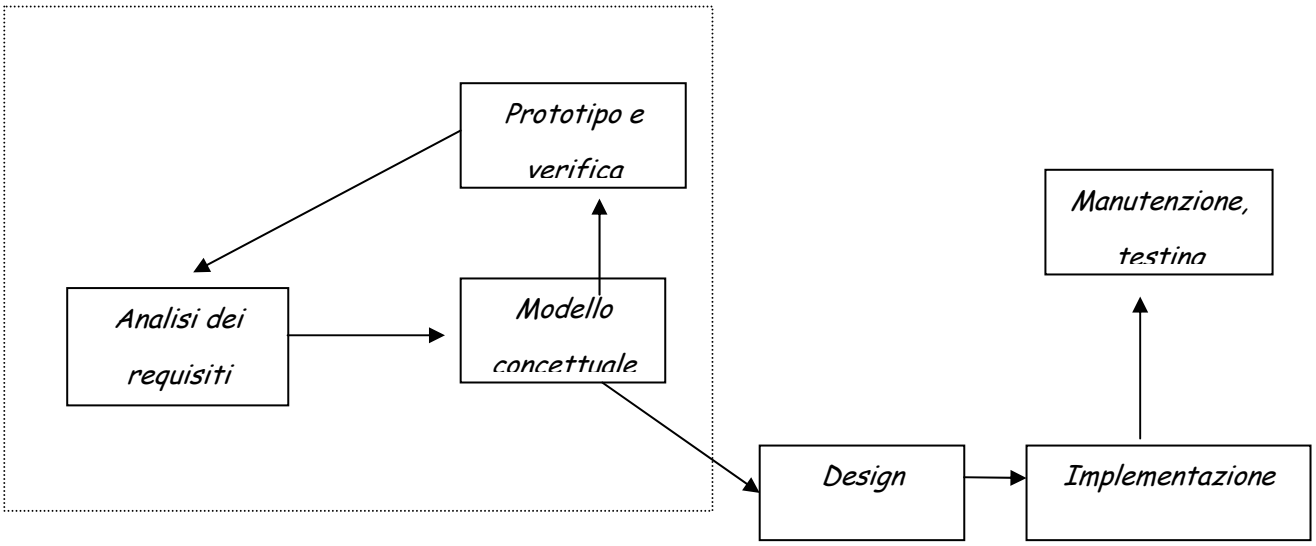


Figura 2 Ciclo di vita di un'applicazione web secondo Fraternali

di installazione sul web nonché, per le applicazioni destinate al world wide web, la richiesta di un dominio. Il testing è fondamentale durante tutto il ciclo di vita ed è il punto di partenza per un'applicazione di qualità

Gli altri modelli proposti sono molto simili a quello descritto, che eredita molto dai modelli tradizionali di Ingegneria del software.
Rispetto alle tecniche tradizionali di sviluppo di software si può notare una maggiore importanza dell'utente nelle scelte progettuali. Infatti, poiché un'applicazione sul web è rivolta ad un vasto pubblico, c'è bisogno innanzitutto di dare un profilo di riferimento alle varie classi di utenti. La stessa fase iniziale di analisi dei requisiti dovrebbe essere eseguita avendo come riferimento i vari profili di utenti dell'applicazione. Ad esempio si può distinguere tra l'utente esperto del dominio che accede direttamente alla funzionalità richiesta e quello meno esperto che va guidato con un sistema di help-on line efficiente.
Poiché un'applicazione web si presenta all'utente con un insieme di pagine web collegate tra di loro mediante link diretti o indiretti, un altro aspetto importante in fase progettuale consiste nello stabilire i percorsi possibili di navigazione associandoli ai profili d'utente. Un utente potrebbe raggiungere l'obiettivo in più modi, lo scopo è quello di ottimizzare la navigazione.

Una metodologia orientata all'utente è presentata in un articolo di Gnaho e Larcher [4] in cui la costruzione di una applicazione web è strutturata come un processo di sei passi:

1) "*Modeling the Potential Users Goals*": definire gli obiettivi che possono essere raggiunti dagli utenti che usano la Web application. A questo scopo si distingue tra obiettivi "informazionali", che implicano accesso ai dati in lettura , e obiettivi "applicativi", che implicano interazione con il sistema e modifica dei dati.
2) "*Modeling the Types of Potential Users*": definire i profili degli utenti che useranno la web application;
3) "*Modeling the WA Information Domain*": definire il modello concettuale dei dati che verranno trattati dalla Web application. Si distingue tra i dati strutturati presentati come oggetti all'interno dell'applicazione di cui vengono definiti attributi, metodi e relazioni, e i dati non strutturati che si riferiscono a informazioni descrittive che non constituiscono il nucleo della web application;
4) "*The Web Navigation Conceptual Modeling*": definire il modello di navigazione degli utenti coinvolti, come un insieme di navigational semantic unit che

rappresentano i vari modi con cui un utente può raggiungere un determinato

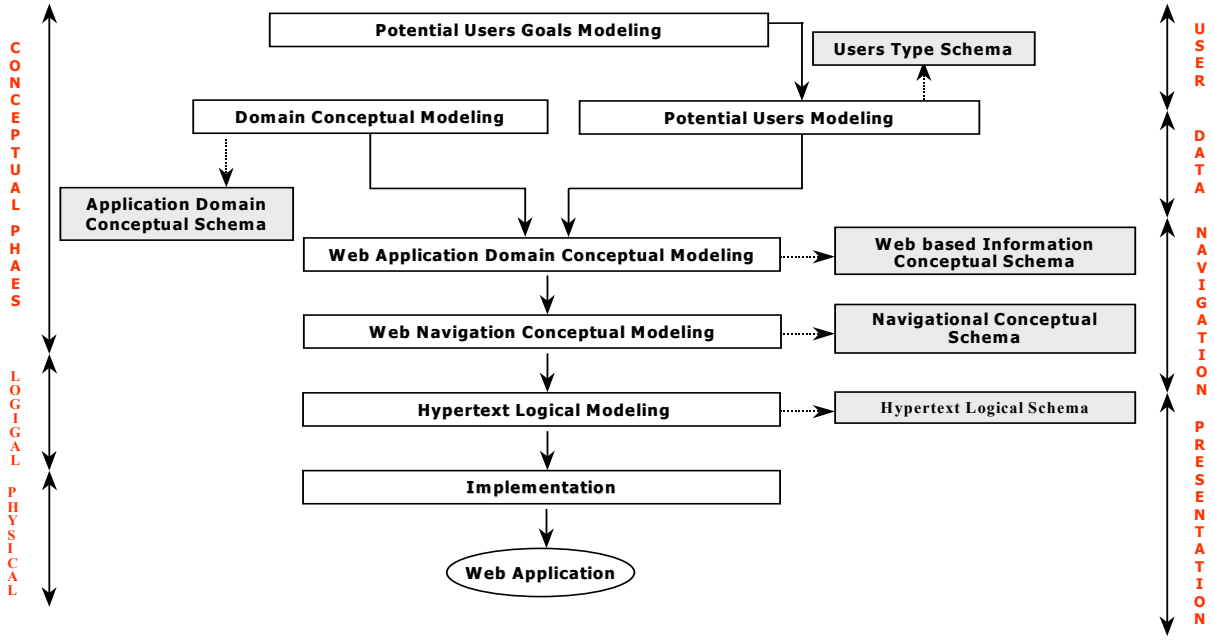


Figura 3 Ciclo di sviluppo presentato da Gnaho e Larcher

obiettivo. In questa fase ognuno di questi percorsi è suddiviso in una serie di nodi di navigazione, che potranno essere promossi a pagine, e di azioni elaborative, che potranno essere applicazioni. La struttura di un NSU è composta da un insieme di componenti detti "Ways of Navigating" (WoN). Un WoN rappresenta un percorso di navigazione associato ad un insieme di utenti a cui corrisponde un determinato profilo. La struttura di una WoN è composta da Nodi connessi da Link di navigazione.

5) "*The Web Hypertext Logical Modeling*": trasformare i modelli di navigazione in modelli logici che verranno poi implementati con la particolare tecnologia utilizzata.
6) "*The Implementation of the WA*": implementare la web application mappando il modello logico nella tecnologia realizzativa scelta. E' solo in questa fase che si decideranno i tool e gli ambienti che verranno utilizzati che potranno essere differenti per le varie parti del sistema.

La figura 3 illustra i passi appena descritti.

Si noti come ad ogni passo descritto è associato uno schema rappresentativo dell'astrazione di livello opportuno dell'applicazione.

CAPITOLO 2 - MODELLI ARCHITETTURALI

1 Modelli architetturali

Una prima classificazione delle applicazioni sul web la si può ottenere facendo riferimento ai tipici modelli architetturali di alto livello con cui sono realizzate. Per fare ciò non ci si concentrerà sulla particolare tecnologia che rende possibile tali modelli, ma verrà definita l'architettura del sistema come una visione di alto livello dei componenti che costituiscono l'applicazione e delle relazioni tra essi.
Una web application non può prescindere dai seguenti componenti:

- Un browser client che interpreta un linguaggio ipertestuale (HTML,XML)
- Un Web server che accetta le richieste per le pagine web
- Un application server che implementa la dinamica dell'applicazione.

Le prime applicazioni distribuite prevedevano un'architettura client-server con il nucleo dell'applicazione localizzato principalmente sul server. Questa categoria può essere classificata come *modello "Thin Web Client"* visto che il client è "leggero", poiché privo di controllo sull'applicazione.
Con lo sviluppo di nuove tecnologie e nuovi linguaggi di script si è passati a un modello dove il client è meno inattivo, e partecipa alla dinamica dell'applicazione: *modello "Thick Web Client"*.
Questo è stato l'inizio verso un tipo di applicazione completamente distribuita in cui intervengono altri protocolli oltre l'HTTP dove il browser Web serve come interfaccia per l'applicazione.
Nei prossimi paragrafi verranno trattati, in dettaglio, i vari modelli.

1.1 Thin Web Client

L'applicazione ha il suo nucleo sul server, mentre il client ha un ruolo passivo.
Il client invia una richiesta al server, che la esegue ed invia una risposta. In questo caso il client è il browser che è eseguito sulla macchina client e il server è un web server che comunica con il protocollo HTTP.
La seguente figura mostra una vista di alto livello del modello.

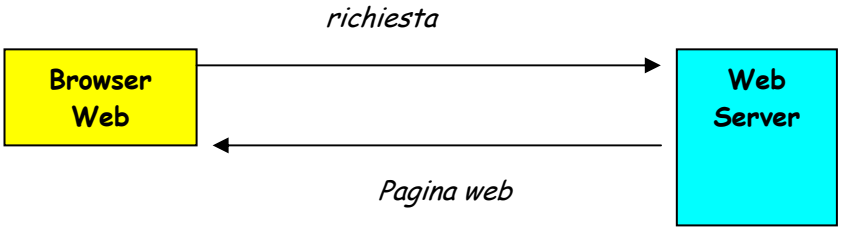


Figura 1 – Vista di alto livello del modello Thin Web Client

Sul server si individua una parte che accetta le richieste (il web server) e una parte che le processa (application server).
La dinamica dell'applicazione è eseguita sul server in seguito a una richiesta che invoca una particolare risorsa, in genere un'altra pagina che include script eseguibili da un interprete situato sull'application server.

La figura seguente mostra come viene costruita dai componenti la pagina web di risposta al client.

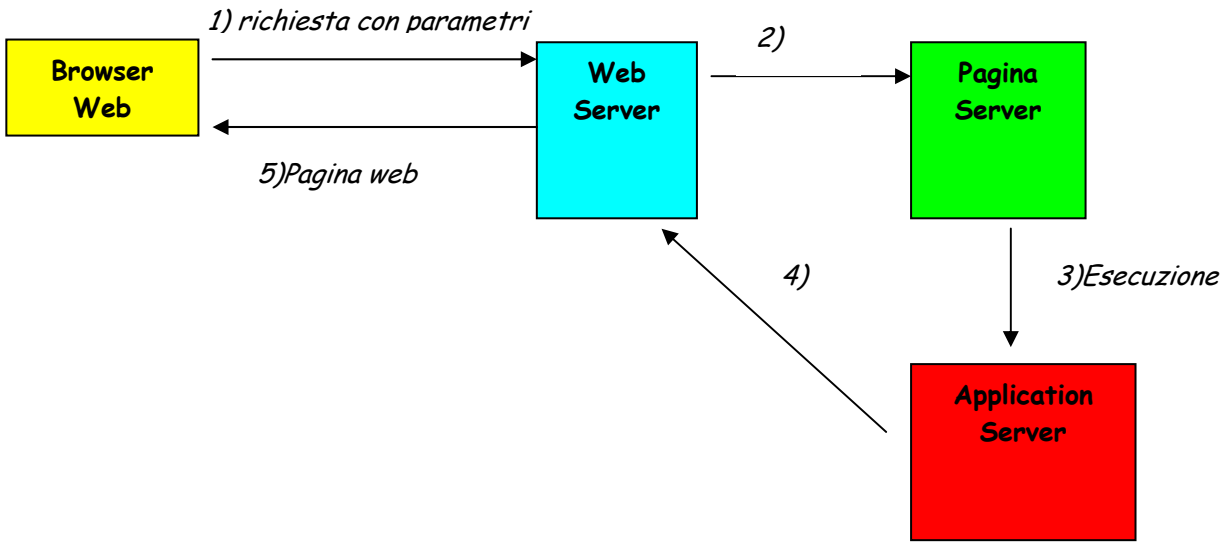


Figura 2 – Interazione tra client e server nell'ambito del modello architetturale Thin Web Client

La distinzione tra web server e application server è puramente concettuale: in pratica l'application server è la parte di web server che contiene l'interprete per l'esecuzione della pagina server.

Un'architettura di questo tipo è anche detta 3 *tier architecture* visto che sono 3 i componenti principali che partecipano all'applicazione.

Esaminando più dettagliatamente, si può dividere l'application server in un componente dedicato alla semplice interpretazione delle pagine server e altri componenti che individuano il tipo di risorsa con cui tali pagine interagiscono. Per la maggior parte delle applicazioni tale interazione avviene con un database che viene interrogato dalla pagina server. Il database viene però interfacciato con un oggetto che ne consente la gestione e che mette a disposizione metodi per interrogazioni e modifiche.

Si può, quindi, generalizzare con la seguente figura e dire che l'application server interagisce con oggetti locali o remoti. Come avviene l'interazione non è lo scopo di questa fase, ma possiamo si può anticipare che, oltre al protocollo http, esistono protocolli proprietari e non (DCOM, Java RMI, ORB..) che consentono la comunicazione tra oggetti remoti.

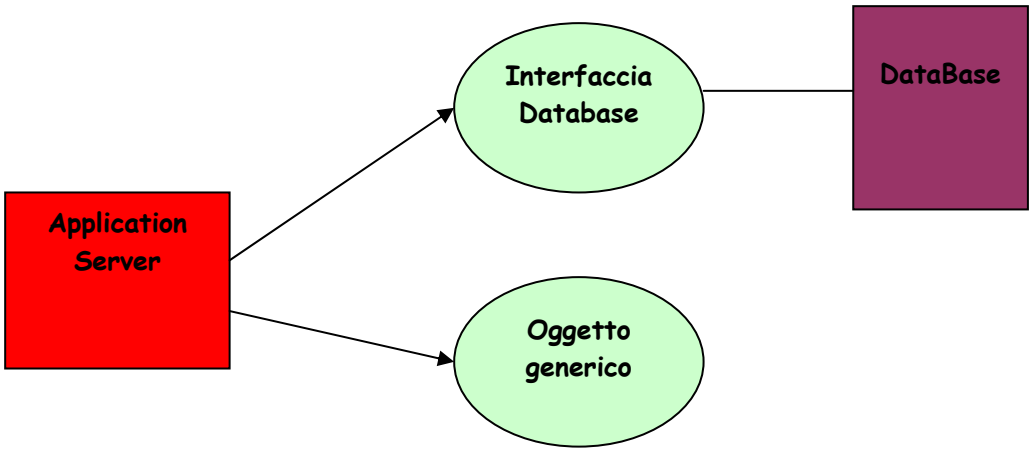


Figura 3 – Interazione dell'application server con oggetti esterni all'applicazione

1.2 Thick Web Client

Con tale modello architetturale si estende il modello precedentemente analizzato aggiungendo attività anche dal lato client. L'obiettivo è quindi quello di non appesantire il server per aumentare le prestazioni dell'applicazione. Il client è rappresentato dal browser che interpreta il linguaggio ipertestuale che costituisce la pagina web. Con l'introduzione di nuove tecnologie implementative il linguaggio HTML è stato arricchito di nuovi componenti che possono consentire interazione con l'utente. In pratica anche nel browser vi è un motore interno che interpreta gli script presenti nella pagina web (ad esempio scritti in Javascript) e tale motore può interagire con oggetti presenti sul client. Caricando la pagina web il browser "scarica" gli oggetti necessari all'esecuzione della pagina (Java applet, controlli ActiveX) e interagisce con questi.

La figura seguente riassume i componenti interessati. Il server è stato indicato in maniera generica, visto che lo schema è quello visto nel paragrafo precedente.

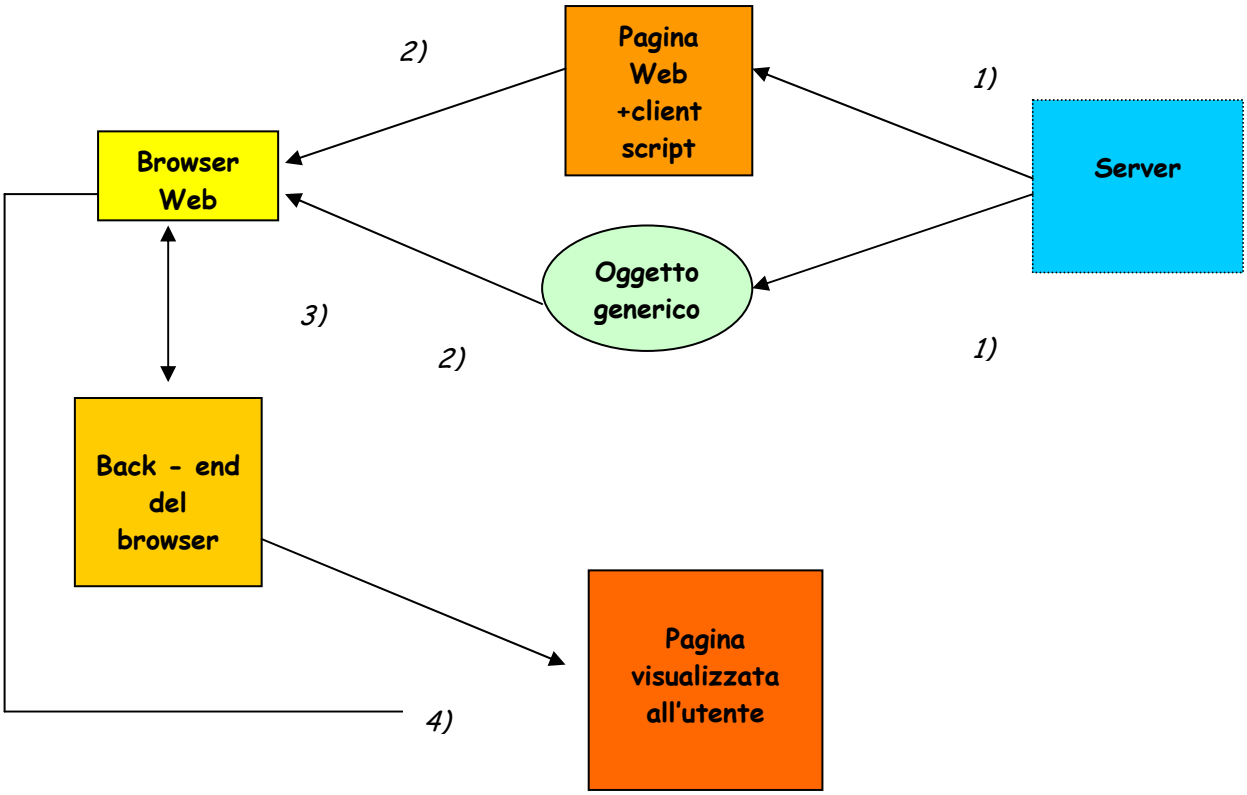


Figura 4 – Vista di alto livello del modello Thick Web Client

CAPITOLO 3 - TECNOLOGIE REALIZZATIVE

1 Introduzione

Il panorama delle tecnologie a supporto dei modelli architetturali visti è abbastanza ampio. Motivi di efficienza e di semplicità fanno sì che esistano tutta una serie di linguaggi e software che riescano a soddisfare le esigenze delle varie tipologie di sviluppatori. Si passa così da semplici linguaggi di scripting, abbastanza poveri nelle funzionalità ma semplici ed intuitivi, come possono essere VBScript e Jscript dal lato server, oppure JavaScript dal lato client, fino a linguaggi più robusti e completi, quali PERL e Java. I linguaggi di scripting si distinguono anche per essere dei linguaggi interpretati e non compilati: questa scelta impone ulteriori vincoli sulla piattaforma software che fa da interprete e rende più complesso il problema di garantire la sicurezza degli script stessi. Un altro fattore che contribuisce a variegare il panorama è dato come al solito dal fatto che alcune di queste tecnologie sono proprietarie: in generale quelle che nascono (e si appoggiano preferibilmente) ad un ambiente Unix sono tecnologie freeware, mentre quelle che si appoggiano all'ambiente Windows sono proprietarie. D'altronde queste ultime garantiscono sempre la compatibilità e interoperabilità con tutti gli altri ambienti e pacchetti Microsoft, che sono attualmente di gran lunga i più utilizzati dagli utenti meno esperti.

Si premette comunque che nell'architettura di un'applicazione web possono convivere due o più di queste tecnologie in modo da sfruttare laddove possibile tutte le peculiarità rispettive. Si cercherà pertanto di descriverle separatamente o mediante gli accoppiamenti più naturali o più diffusi, ferma restando la possibilità di arrivare ad implementazioni ibride.

Ecco quindi una rassegna delle principali tecnologie, in modo critico e con un occhio all'architettura implementata.

Dopo le tecnologie, ci sarà anche una breve rassegna dei linguaggi ipertestuali sui quali tali tecnologie si basano. Il linguaggio HTML, a tutt'oggi decisamente il più utilizzato, è molto semplice e limitato, per cui si configura come una base d'appoggio rigida per i linguaggi sovrastanti. Viceversa i linguaggi basati sullo standard SGML (ed in particolare l'implementazione finalmente supportata da alcuni browser, XML) iniziano ad essere dei seri strumenti di modellazione dei dati, le cui funzionalità debbono essere tenute seriamente in conto nell'ambito della progettazione dell'applicazione web.

2 Tecnologia CGI (Common Gateway Interface)

Si tratta probabilmente della prima tecnologia che si è diffusa in maniera invasiva sul World Wide Web ed è perciò da considerarsi come pietra di confronto rispetto a quelle successive, nell'ambito delle architetture client/server.

L'architettura è ispirata al modello "thin web client":

- dal lato client non è necessario alcun supporto particolare, basta un browser HTML che operi con la pagina client ed attivi, ad esempio a seguito dell'invio di un form, il programma CGI presente sul server;
- dal lato server deve esserci un programma apposito che funzioni da interprete (ad esempio Apache Web Server) e il programma CGI che può essere scritto in uno dei linguaggi di scripting noti al Web Server (il più comune è PERL - Processing Estraction Report Language).

Il passaggio dei dati è molto semplice: il client richiama il programma server aggiungendo dei parametri sulla linea di comando (metodo GET) oppure in un pacchetto esterno (metodo POST); il server esegue il programma cgi associato e risponde elaborando una vera e propria pagina HTML, che viene mandata al browser del client.

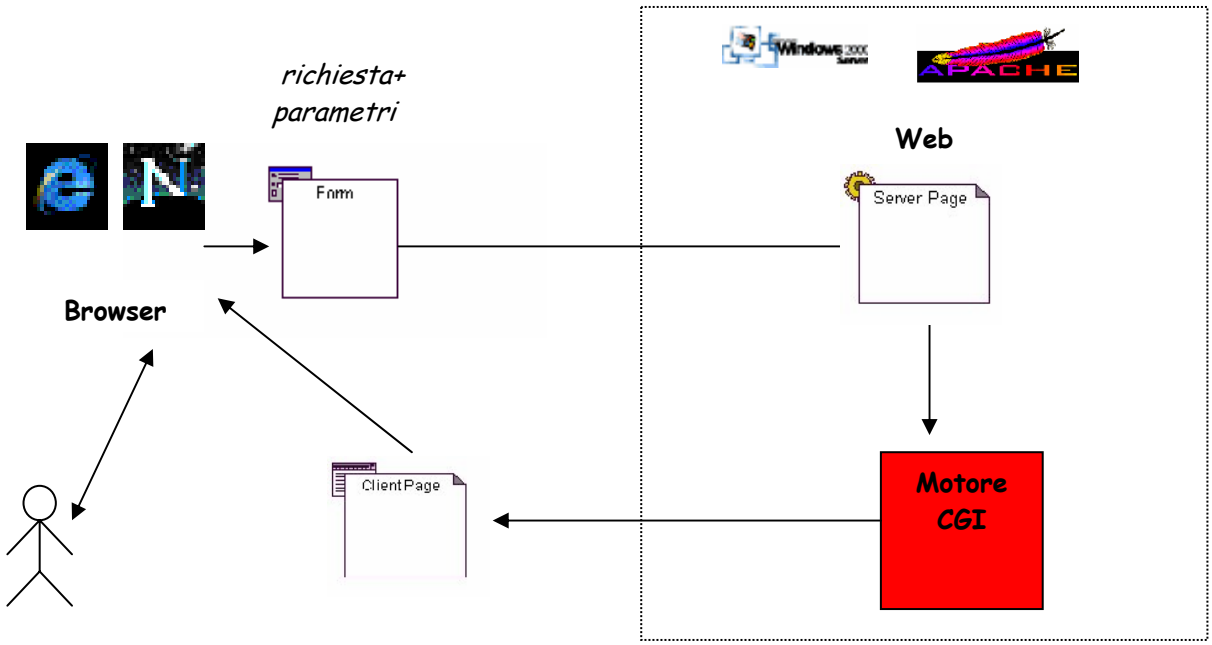


Figura 1 Tecnologia CGI

L'ambiente in cui è stata sviluppata tale tecnologia è quello UNIX, anche se al giorno d'oggi sono state create versioni funzionanti anche per altri ambienti come Windows, etc. Questo fattore è da un lato un vantaggio, dall'altro uno svantaggio. Infatti l'ambiente UNIX è notoriamente non proprietario, così come anche i programmi di Web Server per esso realizzati. Non è qui il luogo di svolgere un confronto tra i sistemi operativi UNIX e Windows, ma è noto come il primo sia più attento alle questioni riguardanti la sicurezza dei dati, conseguenza del fatto che UNIX è nato come sistema operativo per le reti, mentre Windows è un adattamento del sistema operativo DOS, nato per i personal computer. D'altronde, al giorno d'oggi Windows (con tutti i suoi pacchetti applicativi) è nettamente più diffuso di UNIX nei sistemi di piccole dimensioni per cui l'incompatibilità (che pure si sta cercando di colmare) tra CGI e le applicazioni in ambiente Windows fa preferire spesso l'utilizzo di altre tecnologie quali ASP.

Il PERL, che è il principale linguaggio di scripting cui si appoggia CGI, è un linguaggio abbastanza potente e versatile, specialmente per quanto riguarda la gestione delle stringhe (mentre è viceversa da sconsigliare per applicazioni scientifiche). Esso è particolarmente adatto per applicazioni in ambiente UNIX in quanto si interfaccia alla perfezione con le primitive di sistema UNIX. Non segue però il paradigma object oriented e questo ne limita l'utilizzo per applicazioni più complesse e manutenibili per cui la tecnologia CGI-PERL è un po' in declino. Ulteriori svantaggi nell'utilizzo integrale di tale tecnologia derivano dal fatto che l'elaborazione dei dati è sostanzialmente tutta a carico del server, il che porta a problemi di lentezza e scalabilità che ne sconsigliano l'utilizzo per applicazioni complesse.

La comunicazione con i database avviene tramite l'utilizzo di funzioni implementate in moduli aggiuntivi che consentono di realizzare query in SQL. Sono supportati, all'interno del linguaggio PERL, anche le interazioni tramite altri protocolli del livello applicazione (OSI-7), come i protocolli per l'FTP e la posta elettronica.

3 Tecnologia ASP (Active Server Pages)

La seconda tecnologia della quale ci si occupa è quella legata ai file di tipo ASP. L'architettura è abbastanza simile a quella CGI.

Il sistema ASP si basa sull'utilizzo di pagine server (con estensione .asp) le quali rappresentano un'estensione delle tradizionali pagine HTML: infatti un file ASP contiene tag propri di una pagina HTML alternati con righe di codice, scritte in un linguaggio di scripting, le quali possono utilizzare variabili proprie (visibili sul server), variabili del client (mantenute tramite cookies), oppure d'ambiente (ottenute sempre comunicando col browser) L'output di questo codice è dato sostanzialmente da ulteriori righe HTML che vanno ad integrarsi con quelle già contenute nella pagina. L'utente vedrà quindi soltanto una semplice pagina HTML, frutto delle elaborazioni sul lato server.

La possibilità di includere variabili del client è data dall'utilizzo dei cookies. Si tratta di piccoli file che vengono mantenuti sul client e gestiti dal browser, ma che possono anche essere visti dal server. Si tratta di una generalizzazione del trasferimento di dati dal client verso il server. Il programma server può ora interagire con il client formulando delle vere e proprie richieste cui il client risponde automaticamente con un invio di dati. Si può così gestire una vera e propria sessione di comunicazione, che vada a instaurare una comunicazione non stateless tra client e server, il che permette una progettazione più matura ed object oriented, nonché più orientata al parallelismo tra client e server.

Un grosso vantaggio rispetto alla tecnologia CGI è la possibilità di interagire con oggetti esterni richiamandone i metodi forniti con l'interfaccia.

La figura seguente mostra come ASP può essere vista come caso particolare dell'architettura "Thin web client".

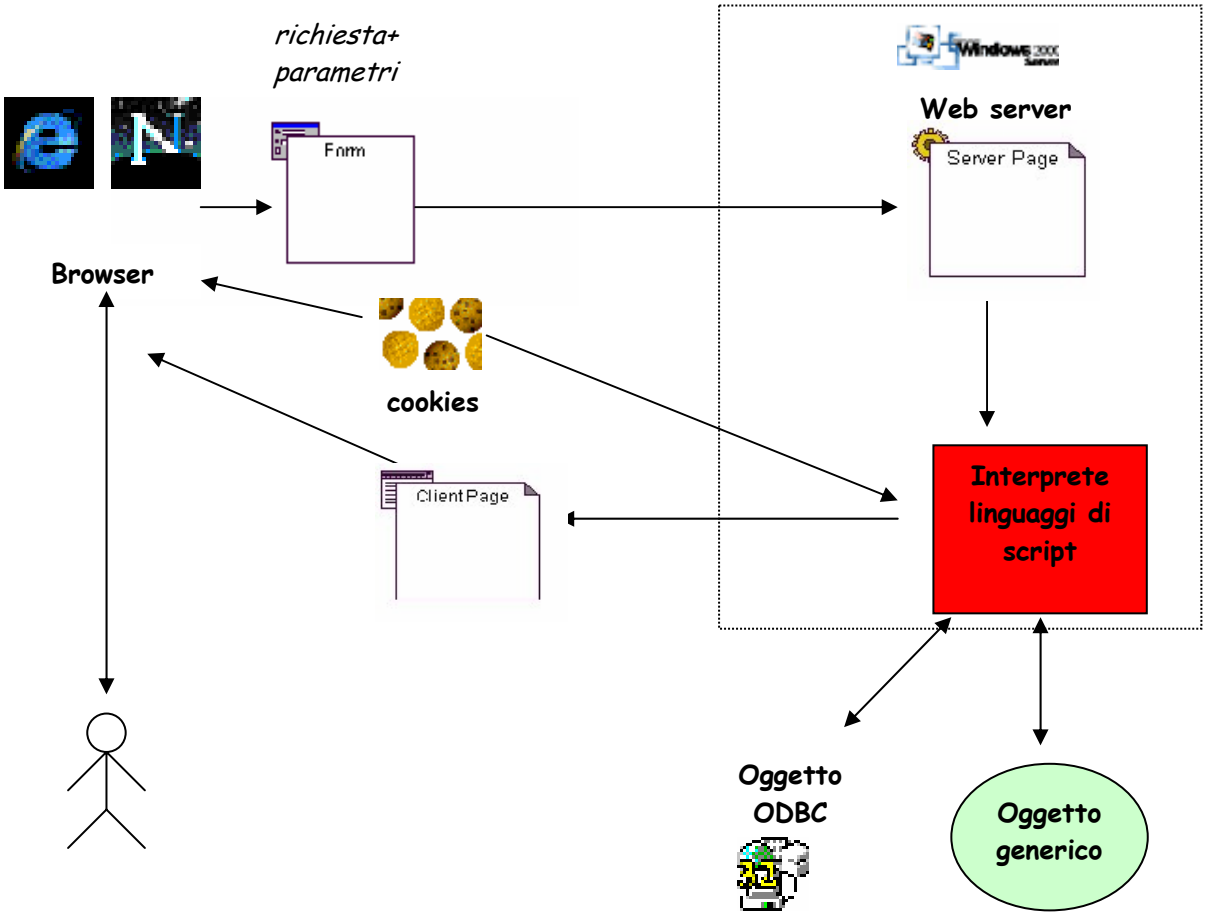


Figura 2 Tecnologia ASP

La tecnologia ASP è proprietaria: il Web Server che rende possibile l'interpretazione degli script è IIS (Internet Information Server) che è a sua volta un programma di proprietà della Microsoft. Ciò condiziona, nel bene e nel male, l'utilizzo della tecnologia. In particolare è ottima l'interazione con qualsiasi altro componente della famiglia Office o Visual Basic, quindi anche con i database Access, mentre necessita di componenti extra l'interazione con altri database. Siccome le applicazioni Microsoft sono più orientate alla semplicità d'utilizzo che all'efficienza e alla sicurezza dei dati, ciò implica che la tecnologia non è in realtà molto affidabile per applicazioni complesse, mentre è indicata per applicazioni di piccole e medie dimensioni, specie per PC.

La tecnologia ASP comprende due linguaggi di scripting predefiniti: VBScript (che si ispira a Visual Basic) e Jscript (che si ispira a Javascript). Si può inoltre

raggiungere la compatibilità, con l'utilizzo di ulteriori add-on al Web Server, con altri linguaggi di scripting come PERL. Pertanto ne risulta semplificato l'utilizzo per la maggior parte dei programmatori di piccoli sistemi, ed è infatti il linguaggio più utilizzato nelle applicazioni web di piccole e medie dimensioni e senza eccessive esigenze di sicurezza.

Esistono alcune classi predefinite le quali sono utilizzabili da ogni linguaggio di script e che segnano l'estensione di ASP rispetto ai normali linguaggi non orientati alle reti. Il linguaggio va potenziato con ulteriori strumenti di interazione che sono propri dell'ambiente Windows e che permettono l'interazione con i database, con le altre applicazioni Microsoft, con la posta elettronica, etc.

ASP permette e consiglia l'utilizzo, anche nella stessa pagina server, di diversi linguaggi di scripting, in modo da facilitare in qualche modo il riutilizzo del codice.

L'interazione degli script ASP con database, posta elettronica ed altri oggetti esterni necessita di apposite librerie di classi, le quali forniscono metodi ad alto livello.

Come filosofia sostanzialmente ASP si configura come un'estensione di Visual Basic sul web: ne vengono favoriti i programmatori inesperti mentre a quelli più esperti vengono nascosti alcuni particolari che possono portare ad una programmazione più efficiente e affidabile. Si tratta infatti ancora di un linguaggio non fortemente tipato; inoltre anche le capacità grafiche non sono molto efficienti in quanto è legato alle primitive di visualizzazione dei browser (cioè quelle utilizzabili in HTML).

4 Tecnologia PHP (Hypertext PreProcessor)

Questa tecnologia non è molto diversa da ASP come modello architetturale e funzionamento, per cui si può passare direttamente al confronto con la precedente.

PHP innanzitutto non è una tecnologia proprietaria: ciò fa sì che sia utilizzabile sulla maggior parte dei Web Server (come ad esempio Apache). Essa è dotata di un proprio linguaggio di scripting a oggetti, con una nutrita libreria di funzioni standard e la possibilità di includere estensioni al linguaggio che permettano, ad esempio, l'interazione con tutti i più diffusi database SQL.

Come ASP, si tratta ancora di un linguaggio non fortemente tipato (molto comoda e intuitiva è la gestione degli array dinamici) e anch'esso ha delle potenzialità grafiche non troppo spinte al di fuori degli oggetti standard dei browser.

Anche con PHP è previsto l'utilizzo dei cookies ed esistono alcune primitive per la crittazione dei dati e la loro autenticazione.

Un ulteriore vantaggio del PHP è che il suo linguaggio di scripting si ispira al notissima linguaggio C (che è a sua volta simile a Java).

In conclusione il maggior pregio di PHP rispetto a ASP è dato dal fatto di non essere una tecnologia proprietaria, il che permette di abbattere significativamente i costi di realizzazione di piccole applicazioni web.

L'unico difetto che si può imputare a questa tecnologia, rispetto ad ASP, è la minore integrabilità nell'ambito di un ambiente Windows.

5 Tecnologia ColdFusion

ColdFusion è una tecnologia server-side molto simile alla tecnologia ASP. Il nucleo è costituito dall'ambiente di programmazione che si basa sul linguaggio CFML (Cold Fusion Markup Language). Tale linguaggio si basa sulla notazione a tag come HTML, ma supporta la potenzialità della tecnologia server.

I tag del linguaggio CFML consentono di interagire con database scritti con la tecnologia di Microsoft Access con semplici query in linguaggio SQL.

Per utilizzare tale tecnologia il web server deve avere un motore interno che possa interpretare pagine scritte in linguaggio CFML (con estensione .cfm). Tali pagine, così come per la tecnologia ASP, possono contenere sia tag HTML che tag CFML che consentono la costruzione della pagina client da presentare all'utente.

La figura seguente descrive sinteticamente il funzionamento di tale tecnologia riferendosi al modello architetturale "thin web client". Da essa si notano subito la completa analogia con l'architettura vista per ASP: cambia solo il linguaggio server ed il web server associato.

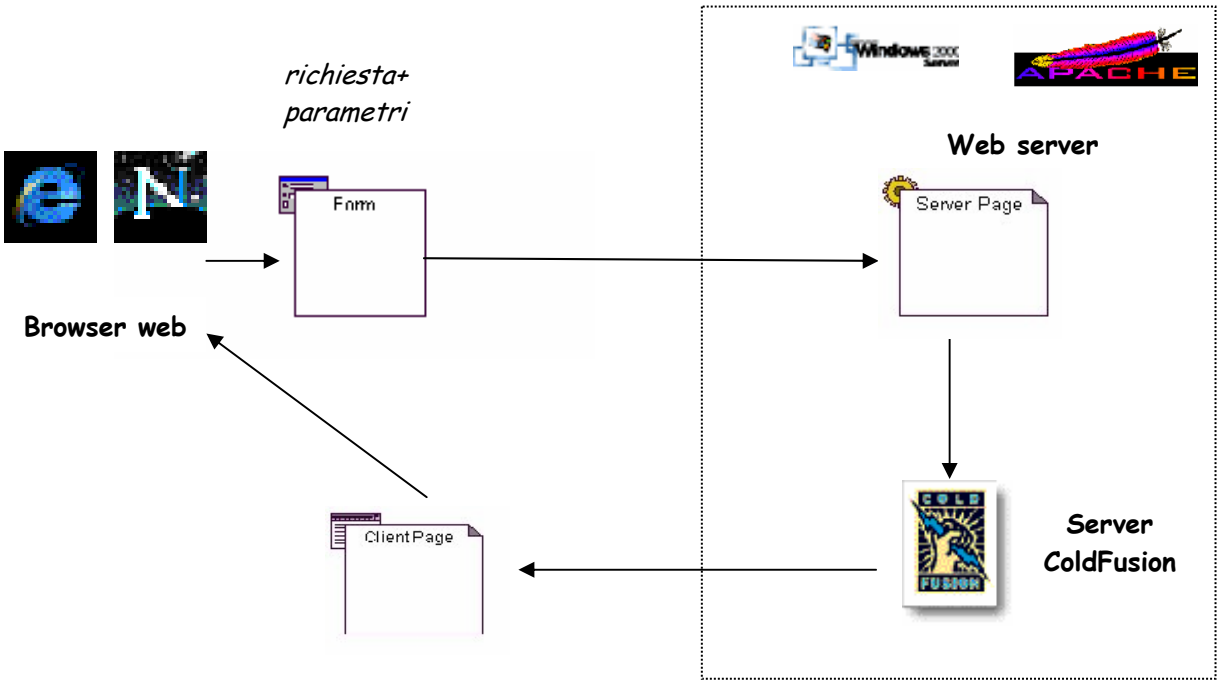


Figura 3 Tecnologia Cold Fusion

6 Tecnologia Javascript

Javascript è un linguaggio che si applica al solo lato client delle applicazioni web.

Dal punto di vista della sintassi esso ricorda perfettamente Java, del quale può essere considerato un fratello minore, ma la similitudine finisce qui in quanto le caratteristiche dei due linguaggi sono pressocchè complementari.

Si tratta in questo caso di un linguaggio interpretato, anche se l'interprete è contenuto in un'estensione, ormai diventata standard, del browser. Javascript è un linguaggio ad oggetti: esso è stato però designato per applicazioni poco complesse, per le quali il modello di gestione della memoria prevede che in esso possano essere create variabili locali ma non classi.

Tali limitazioni consigliano Javascript in due particolari tipologie di utilizzo:

- per creare gradevoli effetti grafici, che migliorino la leggibilità della pagina web, andando anche oltre alle primitive grafiche fornite da HTML, fornendo così alla pagina dinamicità ed interattività (per quanto ciò sia possibile senza scambiare dati col server);
- per svolgere piccole elaborazioni di dati dal lato client, preventivamente alla trasmissione al server, come ad esempio controllo della validità dei valori dei campi in una form, etc.

Nell'ambito di quest'ultimo tipo d'utilizzo, l'esistenza di script dal lato client permette di alleviare il lavoro dal lato server, il che è molto importante per ottenere la scalabilità del sistema (basti pensare ad un sistema con molti client separati che cercano di contendersi le risorse di un piccolo numero di server).

Javascript è anche in grado di gestire, in modo autonomo dall'HTML, forms e cookies, così da poter pilotare l'interazione con l'applicazione lato server.

E' spesso anche sfruttata la possibilità di avere delle applicazioni lato server che restituiscano al browser delle pagine HTML con tanto di codice Javascript, variabile con l'elaborazione, in modo da ottenere "Javascript dinamico".

Sebbene sia in teoria possibile creare applicazioni di dimensioni maggiori in Javascript, ciò è difficile da realizzare e sconsigliabile: tali applicazioni non darebbero infatti garanzie di portabilità e non si presterebbero ad una progettazione matura, impedendo così manutenzione e riuso efficiente.

7 Tecnologia JAVA

Java é un linguaggio di programmazione object oriented, prodotto dalla Sun e liberamente ispirato al linguaggio C++. Esso permette tra l'altro l'implementazione di ogni applicazione tradizionale. In particolare può essere utilizzato nell'ambito della comunicazione client-server.

Java è uno strano linguaggio che é difficile definire compilato quanto interpretato. La "compilazione" di un programma Java fornisce un codice intermedio (bytecode) che può essere trasformato in applicazione eseguibile (ed é questo il caso delle applicazioni Java che vengono eseguite sul server), oppure lasciate in questo codice intermedio che può essere interpretato dai browser (ed é questo il caso delle applet che vengono interpretate sul client). Il risultato che si ottiene é che si può ottenere una comunicazione tra due spezzoni separati di uno stesso programma: un'applicazione (detta anche Servlet) che gira sulla macchina server e un'applet, che si trova inglobata come oggetto di una pagina Web che il client ottiene nel modo tradizionale. Gli obiettivi che riusciamo così a raggiungere sono un bilanciamento di carico tra client e server, sia per quanto riguarda la residenza dei dati che per quanto riguarda l'elaborazione. E' possibile inoltre realizzare una connessione con stato tra i due in quanto la sessione dura quanto l'esecuzione dell'applet (si ipotizza che il server sia sempre in esecuzione).
E' anche pratica molto diffusa l'utilizzo delle applet Java per ottenere pagine client dinamiche, anche molto complesse e con un'interfaccia grafica personalizzabile.
Questa tecnologia si differenzia nettamente dalle altre per essere l'unica a gestire entrambi i lati della comunicazione. Dal lato client è necessaria un'apposita estensione del browser, peraltro ormai standard, detta Java Virtual Machine. Dal lato server invece la situazione è più semplice: si parla infatti di "applicazioni Java", che sono in realtà dei semplici eseguibili.
Utilizzando Java si può ottenere un'indipendenza pressocchè completa dal livello sottostante (HTML), sia dal punto di vista grafico che da quello della comunicazione: non è strettamente necessario utilizzare le classiche strutture di form, né i cookies ed è possibile anche gestire parti significative dell'elaborazione sul client.
Inoltre è possibile una progettazione matura delle applicazioni, che permetta cioè riuso e manutenzione. Anche l'efficienza può essere parzialmente controllata in

quanto abbiamo a che fare con un linguaggio fortemente tipato e con alcune primitive dedicate all'utilizzo efficiente delle risorse.

Java si presta anche alla scalabilità del sistema: infatti, dal lato server, sono previste tutte le primitive di programmazione concorrente, in modo da poter regolamentare l'accesso in modo sicuro ed efficiente.

Java è adatto ad applicazioni web con un certo grado di complessità: applicazioni semplici infatti soffrono sia la maggiore complessità di java come linguaggio di programmazione, che necessita di programmatori più esperti rispetto agli altri linguaggi precedentemente visti, sia il fatto che la necessità di compilare il codice in un bytecode dia vita a dei programmi che sono un po' più pesanti nel caricamento rispetto agli altri, e ciò può dare fastidio agli utenti che abbiano a disposizione una banda più piccola, quelli cioè che utilizzino una semplice connessione telefonica ad un provider.

Delle estensioni come il RMI (Remote method Invocation) consentono a un programma java eseguito su una macchina di invocare un metodo su un oggetto localizzato su una macchina differente garantendo trasparenza al sistema di comunicazione.

8 CORBA

Corba (Common Object Request Broker) è un'architettura proposta dal OMG come base per applicazioni object-oriented distribuite. Il concetto fondamentale è che un oggetto client può utilizzare i metodi di un oggetto server, senza conoscerne l'effettiva localizzazione fisica. In pratica Corba fornisce trasparenza alla localizzazione tra oggetti e ai meccanismi di comunicazione tra essi. Corba è quindi un middleware per realizzare applicazioni distribuite e non un linguaggio di programmazione, come ad esempio Java. Un altro aspetto fondamentale è la possibilità di non dover implementare il tutto con un unico linguaggio di programmazione, ma Corba definisce un linguaggio dichiarativo per la definizione delle interfacce degli oggetti (IDL): Con tale linguaggio un oggetto Corba diventa un'astrazione che possiede un'interfaccia e un'implementazione in un qualsiasi linguaggio di programmazione che riesca a mappare l'interfaccia definita, permettendo così ai vari oggetti di comunicare tra di loro indipendentemente dalle loro implementazioni e localizzazioni.

L'architettura Corba è paragonabile ad un bus hardware: come sul bus interagiscono le periferiche utilizzando un'interfaccia comune, in Corba gli oggetti interagiscono mediante l'interfaccia IDL. I meccanismi di comunicazione sono realizzati mediante l'ORB(Object Request Broker) che è specifico per la particolare piattaforma. Tra di loro gli ORB comunicano utilizzando protocolli comuni.

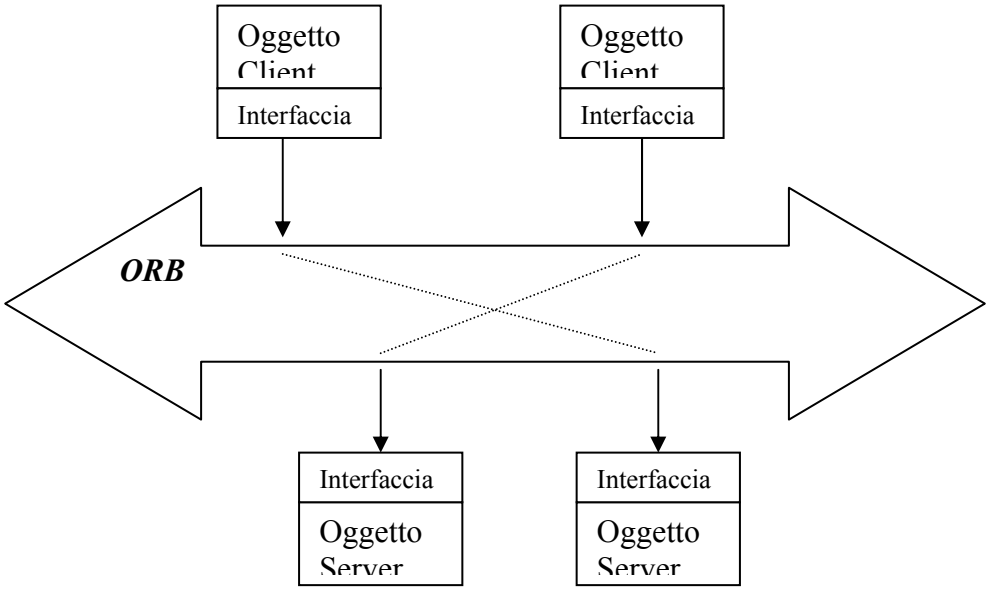


Figura 4 CORBA

9 Linguaggi ipertestuali

Nella trattazione finora fatta delle tecnologie da utilizzare per le applicazioni web si è sempre supposto che tali applicazioni si appoggiassero ad una struttura standard ben nota che regolasse i documenti ipertestuali.

Al giorno d'oggi praticamente tutte le trasmissioni dati che utilizzano il protocollo HTTP si basano sul linguaggio HTML, che è quindi interpretato da ogni browser. Si può quindi affermare, ispirandosi anche alla pila OSI, che il linguaggio HTML costituisca la piattaforma di supporto per i linguaggi utilizzati nelle web application.

Oggi stanno iniziando a diffondersi nuovi linguaggi ipertestuali, che vanno ad incrementare e raffinare le potenzialità dell'HTML, proponendosi quindi come una piattaforma dinamica per lo sviluppo di applicazioni web. A questo scopo si tratterà ora brevemente dei due linguaggi ipertestuali più diffusi, HTML e XML, dando particolare risalto alle differenze tra essi.

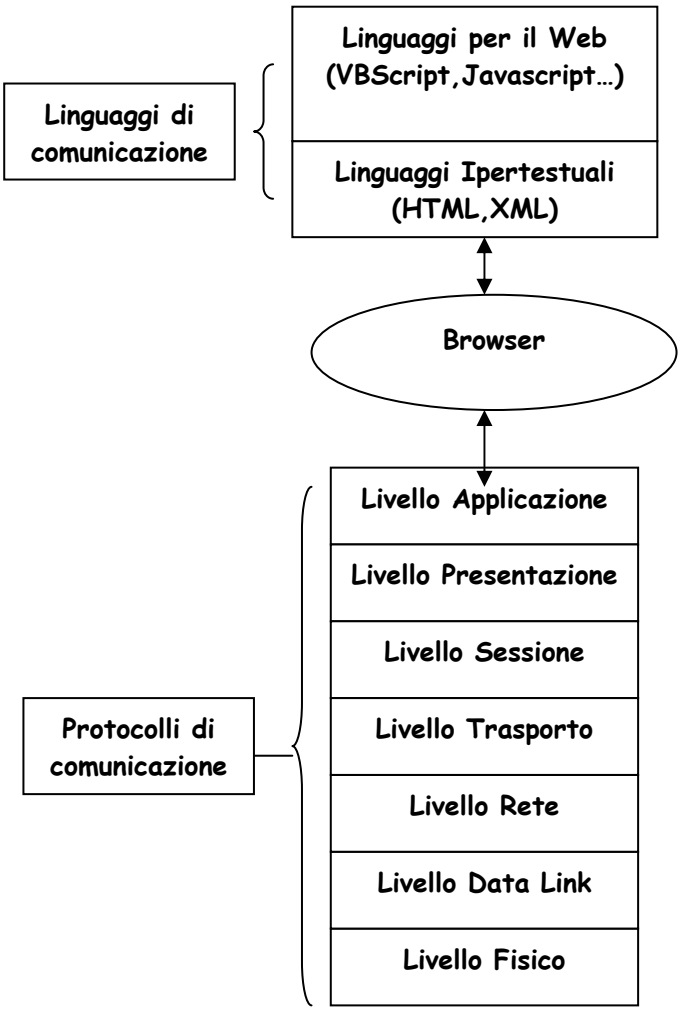


Figura 5

9.1 Linguaggio HTML (HyperTextual Markup Language)

HTML è un linguaggio molto antico (almeno rispetto alle scale temporali del software), che deve la sua longevità appunto al fatto di essere considerato come un elemento costante nello sviluppo di software per web applications. La fortuna, perlomeno iniziale, di HTML è legata alla sua semplicità ed adattabilità.

In HTML risiede del testo più una serie di indicazioni per il browser, racchiuse in tag.

La maggior parte di queste funzioni riguarda la visualizzazione e formattazione del documento ipertestuale: carattere, allineamento, dimensione, etc. Tutto ciò, però, non è in grado di fornire allo sviluppatore informazioni certe sul modo in cui ogni possibile utente visualizzerà le pagine: ad esempio, non è possibile definire la posizione, assoluta o relativa, di un oggetto all'interno di una pagina (come si è visto questo limite viene superato da linguaggi di livello superiore come Java).

L'elemento caratterizzante l'ipertesto è sicuramente la possibilità di dirigersi verso una qualsiasi altra pagina in qualunque punto del World Wide Web tramite il meccanismo dei link. In HTML sono però implementati solo link unidirezionali, i quali sono una struttura che ricorda l'istruzione GO TO dei primi linguaggi di programmazione. Solo tramite artifici nei linguaggi superiori è possibile ricreare strutture di link più complesse, come i link multipli e quelli bidirezionali.

Ulteriori caratteristiche distintive di HTML sono quelle legate alla reperibilità delle pagine nei motori di ricerca, che è essenzialmente legata a dei semplici meta tag, nei quali sono elencate, in maniera arbitraria, le parole chiave della pagina. Ciò rappresenta un problema per i motori di ricerca, i quali, a causa dell'esplosione numerica delle pagine web, non riescono più a garantire una seria valutazione dei risultati di una ricerca, senza andare ad esaminare l'interezza della pagina web.

Il sistema che fornisce HTML per il passaggio dei parametri tra le pagine è quello delle form. Tale sistema, seppur efficiente, è poco personalizzabile, e costituisce un limite nello sviluppo di applicazioni web complesse.

L'unico meccanismo previsto in HTML per separare gli aspetti strutturali da quelli relativi alla visualizzazione, è l'utilizzo dei fogli di stile CSS (Cascade Style Sheet) nei quali possono venirsi a trovare informazioni sulla visualizzazione che, separate dal documento, possono essere riutilizzate in più documenti.

In HTML le informazioni sono, per loro natura, tutte atomiche e ciò impedisce la possibilità di raggrupparle e strutturarle.
Un altro dei più gravi problemi di HTML è l'impossibilità di espanderne le potenzialità. Solo le software house che producono i browser possono in pratica introdurre nuovi tag, oppure associare diverse semantiche ai tag inesistenti. Questa pratica è però sconsigliabile, in quanto si verrebbero così a produrre applicazioni web dipendenti dallo specifico ambiente di utilizzo.

9.2 Linguaggio XML (eXtensible Markup Language)

Nonostante HTML sia di gran lunga il linguaggio ipertestuale più utilizzato, già da tempo si è cercato di teorizzare linguaggi ipertestuali più efficienti ed espressivi.
Lo standard internazionale cui si è giunti è SGML (Standard Generalized Markup Language), del quale l'implementazione più diffusa è XML.
XML è un linguaggio che fornisce informazioni di tipo strutturale e semantico relative ai dati veri e propri. Si ottiene quindi una completa separazione tra struttura dei dati e presentazione. Si apre così tutta una nuova serie di prospettive per le applicazioni web di livello superiore.
L'estensibilità di XML è completa, al punto da poter creare degli ulteriori linguaggi ipertestuali, che siano specializzazioni di XML. Tutto ciò può portare ad una estrema leggibilità del codice.
Questa possibilità di estendere il linguaggio da parte dell'utente implica anche che non è necessaria l'introduzione di ulteriori opzioni da parte del browser, in modo da garantire una compatibilità completa e duratura.
Un altro degli obiettivi di XML è l'estrema leggibilità e chiarezza, la quale implica anche la possibilità di passare in maniera rapida e intuitiva dai modelli di progettazione a quelli implementativi.
Possono esistere quindi dei documenti XML i quali si occupano soltanto di definire un modello delle informazioni che possa essere riusato da altre pagine o in altri progetti: si parla in questo caso di file DTD (Document Type Definition).
Viceversa le informazioni riguardanti la presentazione del documento XML si vengono a trovare in altri file di tipo XSL (eXtensible Style Language), i quali sono a loro volta implementazioni di un altro standard (DSSSL - Document Style Semantics

and Specification Language) e svolgono, sia pure in maniera più puntuale ed estesa, le stesse funzioni dei documenti CSS per HTML.

Come già accennato, la strutturazione dei dati fornita da XML può essere estremamente utile ai motori di ricerca: si può infatti classificare un sito web, anziché in base ad un elenco ordinato di keywords, in base a una precisa gerarchia di argomenti, in modo da ottenere ricerche più efficienti e significative.

La grave limitazione di XML era, fino a poco tempo fa, il fatto di non essere interpretato dai browser più diffusi; dalla versione 5.0 di Microsoft Internet Explorer viene però riconosciuta la prima versione di XML, ed anche Netscape sta adeguandosi.

CAPITOLO 4 - USO DI UML NELLO SVILUPPO DI UN'APPLICAZIONE WEB

1 Introduzione

Le molte analogie tra lo sviluppo di un'applicazione web e di un software tradizionale consentono di utilizzare anche per le prime gli strumenti a supporto del ciclo di sviluppo di un sistema software tradizionale. Uno di questi è lo UML (Unified Modeling Language), un linguaggio standard dell'OMG utilizzato per modellare un sistema software.

In questo capitolo si vuole descrivere come il linguaggio UML può essere utilizzato per rappresentare un'applicazione web.

L'uso del linguaggio UML può essere inserito principalmente in due fasi dello sviluppo di un'applicazione web: analisi e progettazione. Nel seguito è descritto come il linguaggio UML può essere adattato alla nuova tipologia di applicazioni, per meglio descriverne le particolari peculiarità.

1.1 Fase di analisi

Punto di partenza di questa fase è la definizione dei "casi d'uso" del sistema. Così come per un software tradizionale un caso d'uso è caratterizzato da una interazione tra un attore e il sistema o una sua parte. La definizione dei casi d'uso avviene in parallelo alla raccolta dei requisiti con un continuo confronto tra l'utente e l'analista.

Si può dire che in questa fase lo sviluppo di una web application è praticamente uguale a quello di un software tradizionale.

Poiché in genere una web application è rivolta ad un ampio bacino d'utenza, non è facile qui individuare i vari attori del sistema. Ad esempio se si vuole realizzare un motore di ricerca sul web si può distinguere tra un utente esperto che conosce la sintassi del motore e un utente alle prime armi che invece deve essere guidato nella ricerca.

Una volta definiti i casi d'uso del sistema si può iniziare a descrivere mediante gli altri "building blocks" del linguaggio altre astrazioni che si vanno via via dettagliando. In pratica per ogni use case che è stato individuato si inizia a tracciare uno scenario operativo che descrive come l'attore interagirà con il sistema che inizialmente è una scatola nera visto che i dettagli non sono ancora specificati. Per meglio definire tale scenario sono usati i sequence diagram.

Il passo successivo di tale fase è quello di iniziare a dettagliare i vari use case e per ognuno di essi identificare le classi e gli oggetti che ne prendono parte, nonché le associazioni tra esse. In questa fase possiamo utilizzare i class diagram che sono un'astrazione che ha come obiettivo quello di descrivere le classi del sistema e le relationship tra esse (association, dependency, generalization), nonché sequence, collaboration e activity diagram.

I casi d'uso vengono maggiormente dettagliati andando a descrivere anche le possibili associazioni tra essi. Ad esempio un caso d'uso può estendere un comportamento di un altro e ciò viene indicato con la relazione "extends".

Dal dettaglio dei singoli casi d'uso, di sequence e activity diagram può seguire l'introduzione di nuove classi nel sistema.

1.2 Fase di progetto

Anche in tale fase vengono utilizzati i building blocks del linguaggio per rappresentare il sistema da realizzare. L'input di questa fase sono le classi individuate in fase di analisi. Tra esse si possono individuare associazioni del tipo tutto-parti oppure gerarchie che consentono di individuare i sottosistemi che compongono il sistema da sviluppare. Le associazioni sono la base per individuare i messaggi che vengono scambiati tra le classi. Con l'ausilio dei sequence diagram si evidenziano con maggiore dettaglio gli scenari operativi individuati nella fase di analisi.

Per ogni classe si identificano i metodi e gli attributi e in tale fase viene fatta una descrizione dettagliata dei singoli metodi utilizzando ad esempio il formalismo PDL (Program Design Language) ,che consente di mantenere l'indipendenza dal linguaggio di programmazione, oppure gli statechart diagram nel caso in cui il metodo possa essere descritto come un automa a stati.

1.3 Estensioni di UML per le Web application

Si è fin qui data una descrizione generale e poco dettagliata su come utilizzare UML in un progetto software per un'applicazione web. Sono stati utilizzati i termini classe, oggetto, ma chi sono le classi in un'applicazione web? Come si può intuire le regole di UML non sono sufficienti a descrivere tutti i componenti dell'applicazione web, ma c'è bisogno di definire delle "estensioni" apposite che andranno ad integrarsi con

gli elementi di base del linguaggio. Tali estensioni dovranno descrivere i componenti specifici di una web application quali le pagine web, le pagine server…
Una estensione in UML è espressa in termini di: stereotipi, valori etichettati e vincoli.

- Uno *stereotipo* è un'estensione del vocabolario del linguaggio e consente di assegnare un nuovo significato a un elemento del linguaggio.
- Un *valore etichettato* è una nuova proprietà assegnata ad un elemento del linguaggio.
- Un *vincolo* è una estensione alla semantica del linguaggio, specifica le condizioni in base alle quali un modello si può considerare ben formato.

Vengono di seguito descritte le estensioni più significative e per ognuna di esse viene associata un'icona.

Estensioni per le pagine

Una pagina web che compone l'applicazione può essere vista come uno stereotipo di una classe. La distinzione è tra le pagine che risiedono sul server e le pagine client.
Si riportano di seguito le varie tipologie di classi associate a pagine o altri elementi di interesse in un'applicazione web. In una stessa pagina fisica, intesa come il file contenente il codice sorgente, è possibile avere due o più delle seguenti classi:

- *Pagina client*: rappresenta una classe con lo stereotipo <<*client page*>> i cui attributi sono le variabili contenute nella pagina web che hanno come scope l'intera pagina e i cui metodi sono le funzioni che vengono eseguite sul lato client. Tale stereotipo di classe può essere associato ad un componente che viene

Figura 1 Icona associata ad una pagina client

eseguito sul client (Applet java, controllo ActiveX…).

- *Pagina server:*è una classe con lo stereotipo <<*server page*>> con attributi e metodi definiti come per la pagina client e con associazioni anche con altri componenti presenti sul server. I metodi presenti in questa pagina sono associati agli script che vengono eseguiti sul lato server della applicazione

Figura 2 Icona associata ad una pagina server

web

- *Form*: rappresenta una classe separata dalla pagina client e contenuta in essa che serve per interagire con il server. Ogni pagina client può includere più form i cui dati sono inviati a differenti pagine per l'elaborazione Gli attributi sono i campi di input del form, e non ci sono operazioni associate. Si possono definire degli stereotipi anche per tali attributi a seconda del tipo di campo di input associato alla form:
    - Text se il campo è una semplice casella di testo
    - Textarea se il campo è una casella di testo a più linee
    - Select se il campo è una lista tra cui selezionare un elemento
    - Radio se il campo è una lista a selezione multipla
    - Button se il campo è un bottone
    - Reset se il campo è un bottone che fa azzera gli altri campi
    - Submit se il campo è un bottone a cui è associato l'invio dei

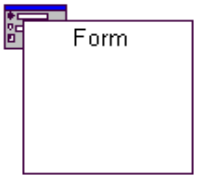


Figura 3 Icona associata ad una form

dati alla pagina che li elabora

- Hidden se il campo è nascosto all'utente

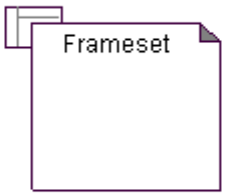


Figura 4 Icona associata ad un frameset

- *Frame:* la presenza di frame in una pagina web consente di avere più pagine web attive contemporaneamente. Puo essere specificata una classe con lo stereotipo <<*frameset*>> che è legata mediante una associazione alle pagine che costituiscono i vari frame dell'output. Infatti in HTML un frame esiste all'interno di un frameset che provvede alla divisione del layout tra i frame

Associazioni tra le pagine

Altre estensioni riguardano le associazioni, vediamo gli stereotipi principali che vengono definiti:

- *Link:* rappresenta un collegamento ipertestuale unidirezionale tra due pagine

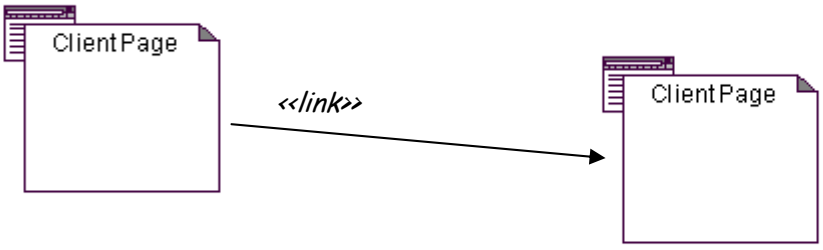


Figura 5 Esempio di associazione di tipo link

web. La direzione del collegamento è indicata dalla direzione della freccia associata allo stereotipo.

- *Submit:* rappresenta un'associazione tra una form e la pagina server che processerà i parametri associati ad essa.

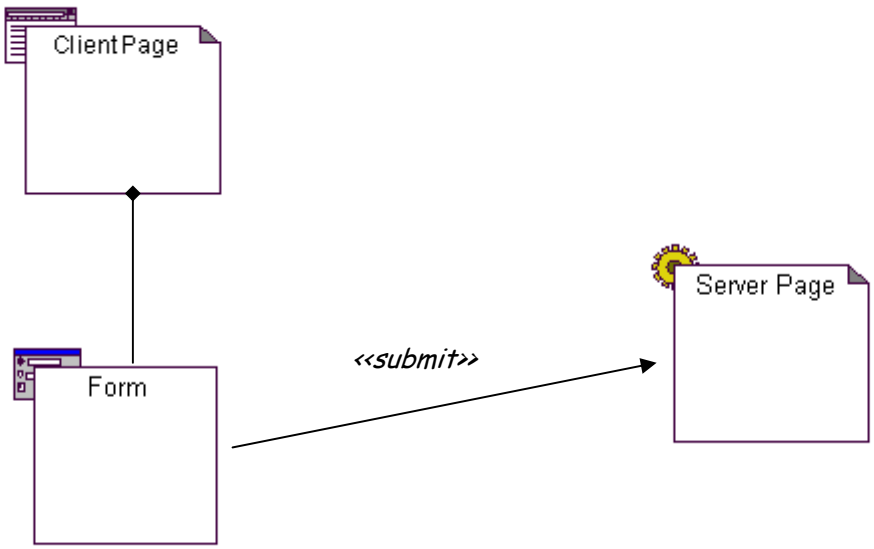


Figura 6 Esempio di associazione di tipo submit

Figura 7 Esempio di associazione di tibo builds

- *Builds:* è una associazione tra una pagina server e una pagina client che è ottenuta dalla prima.

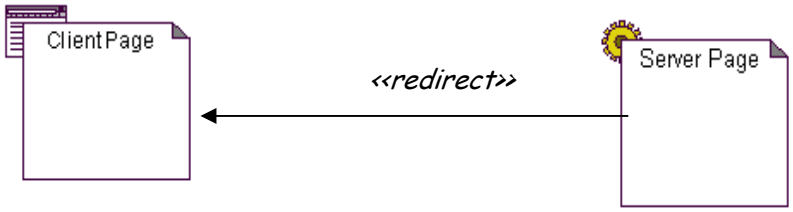


Figura 8 Esempio di associazione di tibo redirect

- *Redirect:* è un'associazione tra due pagine dovuta alla possibilità da parte di una pagina sorgente contenente uno script di passare l'elaborazione ad un'altra pagina.

1.4 Diagramma dei collegamenti

Utilizzando le estensioni definite e le icone associate ad esse si può definire un diagramma dell'applicazione web il cui scopo è quello di evidenziare i collegamenti esistenti tra le pagine del sistema. Non vi è nessun formalismo che obbliga a tracciare il diagramma in un modo particolare, sarebbe opportuno comunque andare a definire i collegamenti tra le pagine che sono incluse in un caso d'uso.

Il diagramma ha la forma di un grafo i cui nodi sono le icone precedentemente presentate, associate alle estensioni (pagine server, client, form, frameset) che rappresentano i componenti che partecipano al caso d'uso in esame, e i cui archi sono i collegamenti tra tali componenti. Per ogni collegamento è associato uno stereotipo che ne indica il tipo.

Il diagramma si può posizionare nella fase iniziale di progetto dell'applicazione web in modo da fornire un'anticipazione del più formale class diagram di UML. A differenza del class diagram, nel diagramma dei collegamenti non vengono riportati gli oggetti che si interfacciano con le pagine dell'applicazione web in esame, ma solo componenti che sono tipici delle applicazioni web. In questo modo il diagramma può aiutare ad individuare un percorso di navigazione nell'applicazione web in esame.

Nella figura seguente viene presentato un semplice esempio in cui una pagina client, contenente una form , ha un link verso un'altra pagina.

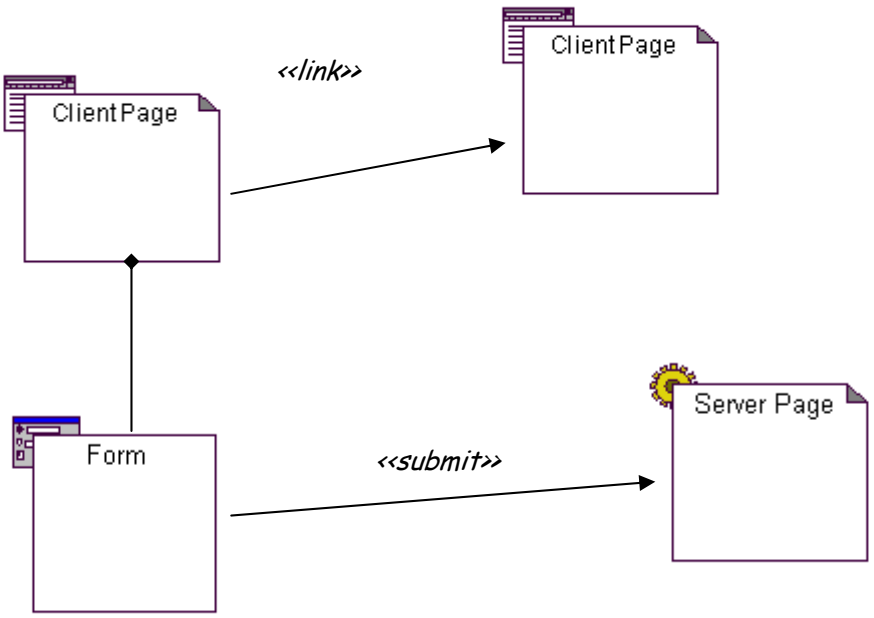


Figura 9 esempio di diagramma dei collegamenti

1.5 Utilizzo dei diagrammi UML per la rappresentazione del modello di un'applicazione web

Con le estensioni appena definite si può rappresentare una web application utilizzando i diagram di UML. Bisogna distinguere tra i diagram che danno una vista statica del sistema (class, component , object, deployment e use case diagram) e i diagram che danno una vista dinamica del sistema (sequence, state, collaboration e activity diagram). In tali diagrammi vanno inseriti i componenti che costituiscono il sistema, siano essi elementi base del linguaggio oppure estensioni.
Vediamo in dettaglio come i diagram si adattano a rappresentare una applicazione completa.

Use case diagram

Dalla fase di analisi e dalla individuazione degli use case, con tali diagram possiamo indicare quali sono le relazioni tra gli use case e gli attori del sistema.
Il diagram può essere presentato a vari livelli di dettaglio, a partire da un use case fino a scomporlo nei suoi componenti. La stesura di tali diagram non subisce modifiche rispetto al caso dello sviluppo di un tradizionale software. Come già detto lo scopo di tali diagram è di definire i modi di utilizzo del sistema da parte degli utenti.

Class diagram

Tale diagram fornisce la vista statica del sistema. In esso sono specificate tutte le classi che costituiscono il sistema, comprese quelle derivanti dalle nuove estensioni prima descritte. Si specificano le associazioni tra esse, che possono essere le associazioni originarie del linguaggio o le opportune estensioni. Fanno parte di tale diagram le molteplici pagine che costituiscono il sistema e le altre classi con cui tali pagine interagiscono. In questo caso la pagina è intesa come classe e l'associazione con la pagina fisica è evidenziata nel component diagram descritto in seguito.
unico component che è appunto il file sorgente.

Object diagram

Da come è stata introdotta la rappresentazione delle applicazioni web, è difficile parlare di oggetti in esse. Si è scelto di associare ad ogni pagina fisicamente accessibile una classe. Allorchè un utente richiede una pagina web client, il web

server ne invia una copia al browser client e si può considerare quest'ultima copia come un oggetto istanziato dalla classe associata alla pagina. Nel caso in cui l'utente richieda l'elaborazione di una pagina server, il web server invia al browser solo i risultati dell'elaborazione e quindi viene istanziato solo l'oggetto associato alla classe con lo stereotipo "pagina client costruita".

Component diagram

I component della web application vanno individuati tra le pagine web fisiche (con estensione .html, .asp, .cgi …) siano esse server o client e i moduli che contengono oggetti o librerie che vengono utilizzati nel sistema. Un singolo component può contenere più classi e tale relazione va specificata nel diagram. Ad esempio una pagina server può costruire una pagina client per presentare informazioni all'utente e in questo caso si hanno due classi associate alle pagine che sono contenute in un

Deployment diagram

Mostrano le configurazioni dei vari nodi elaborativi interessati e dei componenti in essi ubicati. Sono una vista del sistema importante in questo tipo di applicazioni distribuite, perché contribuiscono a capire dove vengono eseguite le varie fasi elaborative ovvero dove andare a disporre le varie risorse software(pagine).

Sequence diagram

Con tali diagram si passa a una vista dinamica del sistema. In genere si specifica almeno un sequence diagram per ogni caso d'uso. Viene dettagliata l'interazione che avviene tra l'attore e il sistema, visto con le classi che lo costituiscono. Nel caso delle web application tali diagram sono particolarmente significativi, visto che possono specificare come viene costruita la pagina per l'utente e inoltre per ogni caso d'uso indicano anche i passi che l'utente deve seguire per giungere al risultato. Si definiscono e descrivono le varie interazioni tra gli oggetti componenti una singola pagina e/o più pagine.

Collaboration diagram

Esprimono la stessa semantica dei sequence diagram, ma con una rappresentazione diversa. Infatti vengono focalizzate le classi che interagiscono con lo scambio di messaggi e non la successione temporale di essi.

2 Esempio di sviluppo di una web application con UML

2.1 Introduzione

In tale sezione si vuole presentare un esempio di analisi, progettazione e realizzazione di una semplice applicazione sul web utilizzando UML con le estensioni descritte in precedenza.

2.2 Definizione dei requisiti

Si vuole realizzare un'applicazione per la gestione di una mailing list sul web. Tale applicazione deve gestire un insieme di utenti iscrittisi alla lista fornendo il loro indirizzo e-mail in modo da poter inviare loro messaggi allo scopo, ad esempio, di tenerli aggiornati sulle ultime modifiche del sito web a cui tale applicazione si appoggia. Un utente può quindi iscriversi alla mailing list e l'amministratore del sito web può inviare messaggi agli utenti iscritti.

2.3 Individuazione degli use case

Dalla definizione dei requisiti appena descritta possiamo individuare una classe generale di utenti del sistema che si specializza in utenti esterni ed amministratore. Dai requisiti specificati si possono identificare i seguenti use case:

- Inserimento di un nuovo utente
- Cancellazione di un utente
- Invio di un messaggio

La figura 10 riporta in un use case diagram tali use case e gli attori coinvolti.

2.4 Individuazione delle classi

Poiché l'utente interagisce con il browser e, quindi ,con una pagina client ,possiamo dire che per ogni use case esisterà almeno una pagina client per l'interazione. Tale

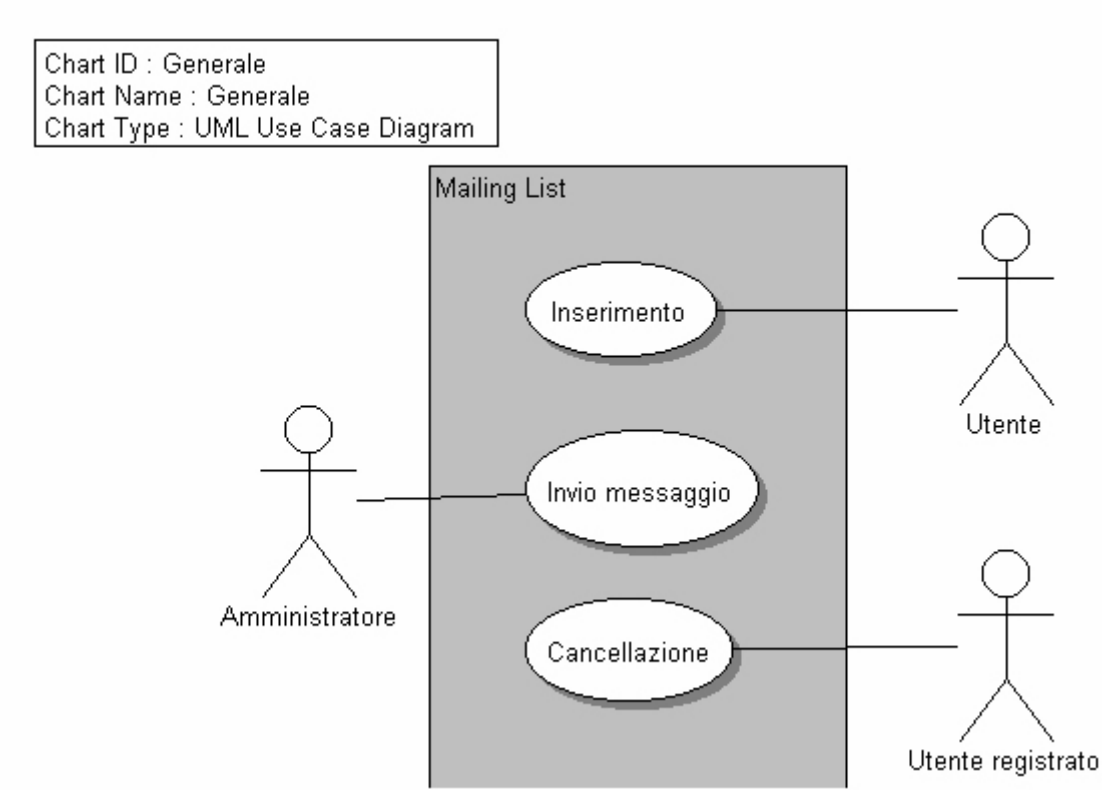


Figura 10 Use case diagram

pagina client può essere "passiva" e dare soltanto informazioni all'utente, oppure può consentire l'interazione con l'utente e con una o più pagine server per l'elaborazione dei dati. Si individuano per ogni use case le pagine logiche che dovranno essere realizzate, facendole corrispondere a classi nei diagrammi UML. Sono definite anche le associazioni tra le pagine ed il tutto è rappresentato mediante il diagramma dei collegamenti precedentemente introdotto.

Use case "Inserimento di un nuovo utente"

L'utente interagirà con una pagina client, che viene ad essere la pagina principale dell'applicazione, nella quale inserirà il proprio indirizzo e-mail da aggiungere al database delle e-mail. Per l''inserimento sarà usato un form in cui inserire l'indirizzo e-mail che viene inviato ad una pagina server che provvede

all'inserimento in un apposito Database e ad inviare una e-mail di conferma all'utente. Tale pagina server deve anche indicare all'utente se l'inserimento è avvenuto con successo o meno. Sono state quindi individuate le seguenti classi:

- Home Page <pagina client>
- Nuova e-mail <form>
- Inserimento < pagina server>
- Risultati inserimento <pagina client>

La pagina server non deve inviare nessun risultato all'utente, ma può mostrare la home page.

La figura 11 mostra il diagramma dei collegamenti tra le pagine individuate per l'use case "Inserimento di un nuovo utente".

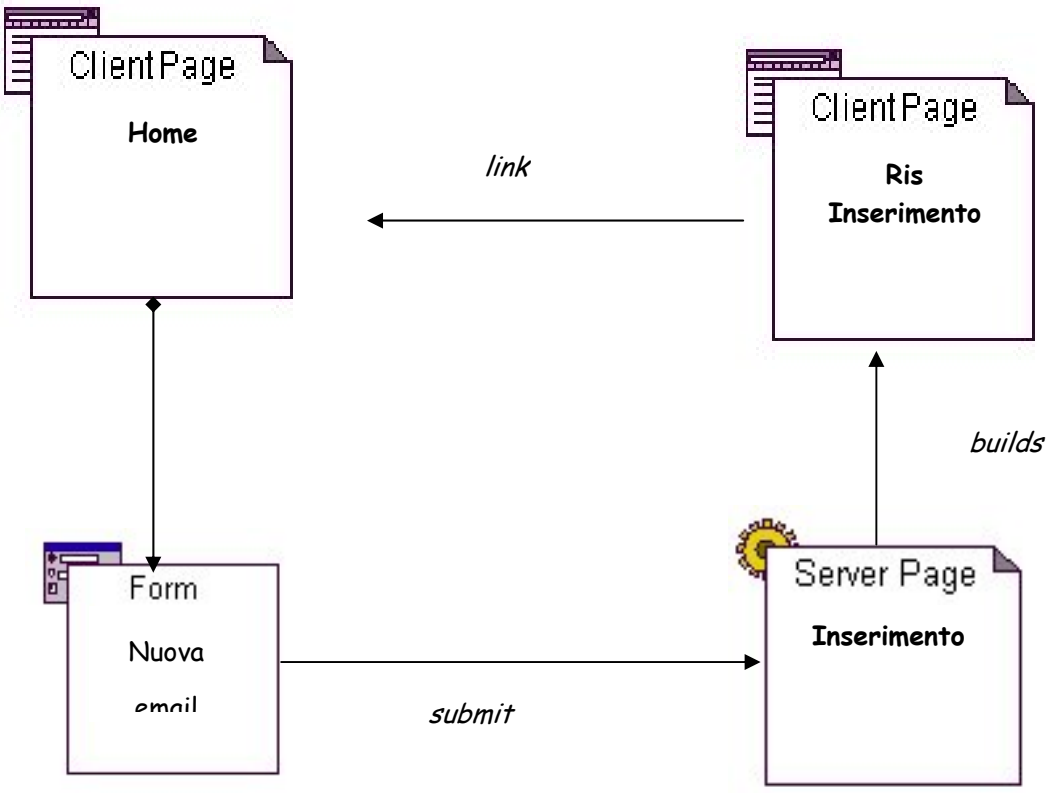


Figura 11 Diagramma dei collegamenti per l'use case "Inserimento"

Use case "Cancellazione"

In questo caso l'utente può richiamare direttamente una pagina server, il cui indirizzo è fornito nella e-mail ricevuta all'atto dell'iscrizione,che provvede ad eliminare la e-mail dal database. Sono individuate le seguenti classi:

- Risultati cancellazione <pagina client>
- Cancellazione <pagina server>

La figura 12 mostra il diagramma dei collegamenti relativo all'use case descritto

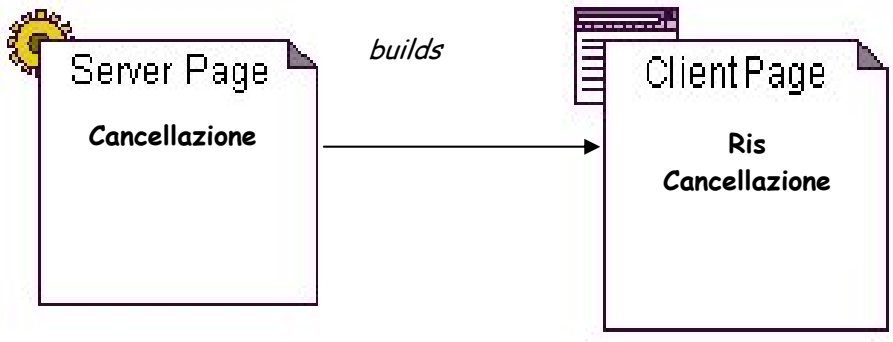


Figura 12 Diagramma dei collegamenti per l'use case Cancellazione

Use case "Invio messaggio"

Questa funzionalità è tipica del solo amministratore. L'amministratore richiama una pagina client in cui deve essere contenuta una form che consente di scrivere il testo del messaggio. L'invio del messaggio a tutti gli utenti iscritti alla mailing list è a carico di una pagina server che deve indicare anche gli utenti a cui è stato inviato il messaggio. Sono individuate le seguenti classi:

- Amministrazione <client page>
- Messaggio <form>
- Amministrazione <server page>
- Elenco destinatari <client page>

La figura 13 mostra il diagramma dei collegamenti tra le classi individuate.

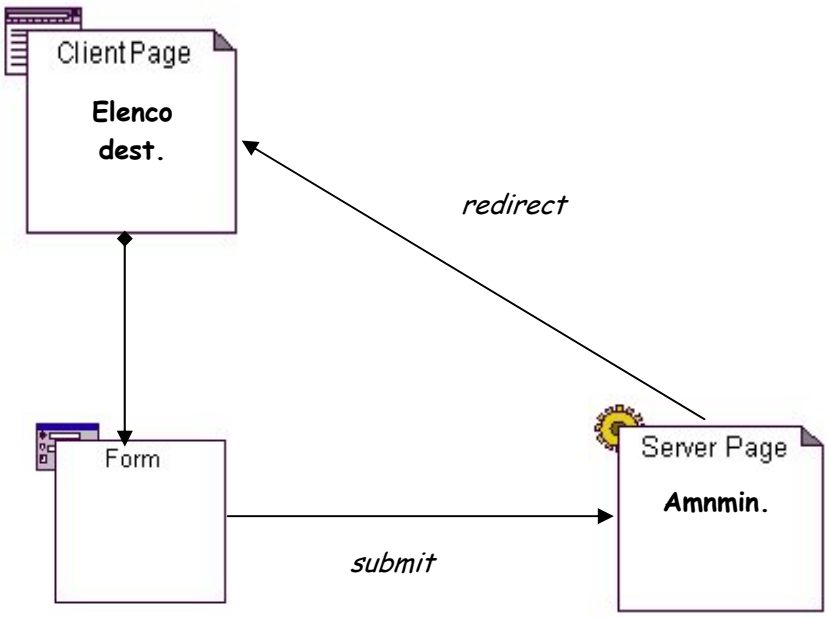


Figura 13 Diagramma dei collegamenti per l'use case Invio messaggio

2.5 Alcune considerazioni di tipo generale

Ogni pagina server individuata, ha un'interazione con il database. Per realizzare tale interazione è necessario che la classe che rappresenta la pagina server interagisca con classi che consentano l'interfacciamento con database. A tal proposito si può anticipare una scelta implementativa: le pagine server verranno implementate con la tecnologia ASP e quindi ogni pagina server interagisce con oggetti predefiniti del linguaggio ASP e con oggetti che consentono l'interfacciamento con il database. Tali oggetti sono istanze di due classi di tipo ADODB: Connection e Recordset. La prima consente la connessione ad un database realizzato in Microsoft Access, la seconda consente di scorrere i record risultanti da una query su tale database. E' poi possibile anche l'interazione con messaggi di posta elettronica. In tal caso vengono istanziati oggetti della classe predefinita CDONTS.Newmail.

Inoltre ciascuna pagina server interagisce con le seguenti classi: Server, Response e Request. La prima serve a gestire variabili di ambiente relative alle singole sessioni create con gli utenti, la seconda serve ad inviare all'utente risposte alle elaborazioni

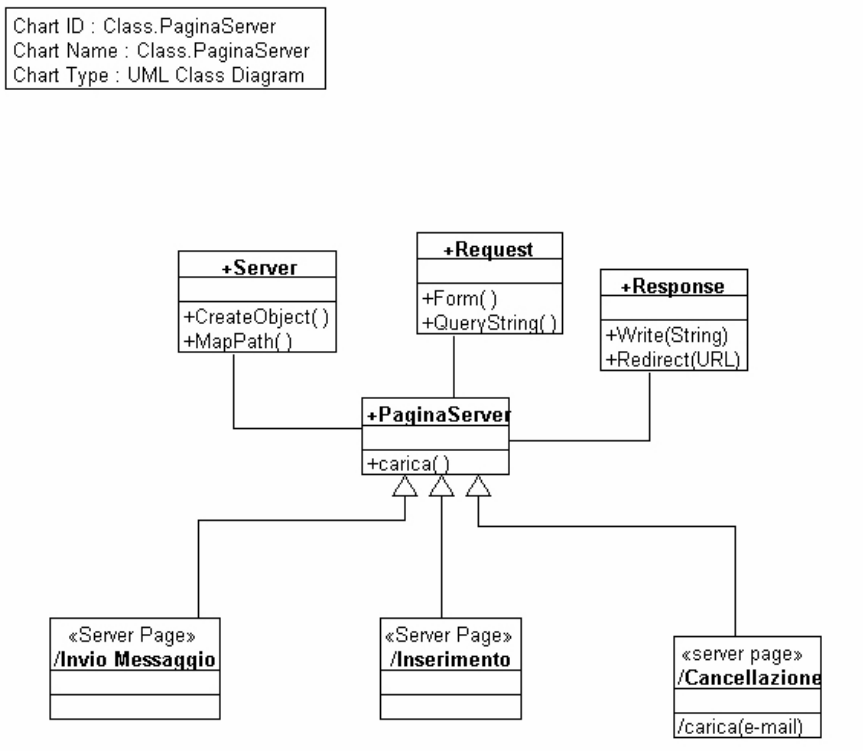


Figura 14 Class diagram delle pagine server

inserite in codice HTML con opportuni tag che verranno interpretati dal browser, la terza serve ad elaborare i parametri provenienti da una richiesta di elaborazione.

In figura 14 è riportato un class diagram che illustra quanto detto, con riferimento all'applicazione in esame. Il diagram mostra una superclasse "Pagina Server" che partecipa ad associazioni con le classi Server, Response e Request. Tale superclasse si può ritenere come padre delle particolari pagine server dell'applicazione in esame. Le pagine server dell'applicazione sono infatti indicate in figura come le figlie della superclasse pagina server.

Il diagramma in figura 14 evidenzia le associazioni di base del linguaggio, nei paragrafi successivi saranno presentati i class diagram che contengono le associazioni tra le singole classi classi.

Sebbene il diagramma è legato al particolare linguaggio implementativo, esso è comunque valido anche quando si utilizza un linguaggio differente a patto di individuare le classi predefinite dello specifico ambiente.

Considerazioni analoghe possono essere fatte per le pagine client, in particolare per quanto riguarda la struttura gen-spec.

Per tutte le pagine client bisogna considerare i seguenti metodi generali:

- Carica: viene richiamato dal browser quando si carica la pagina HTML
- Redirect: viene richiamato su una pagina quando c'è un'istruzione di redirect su un'altra pagina
- Crea: è il metodo mediante il quale una pagina server costruisce una pagina client.

In base a tali metodi si può creare una gerarchia che comprende anche le form, visto che queste sono classi contenute in una pagina client.

Il class diagram in figura 15 illustra le relazioni le relazioni di base tra le pagine client.

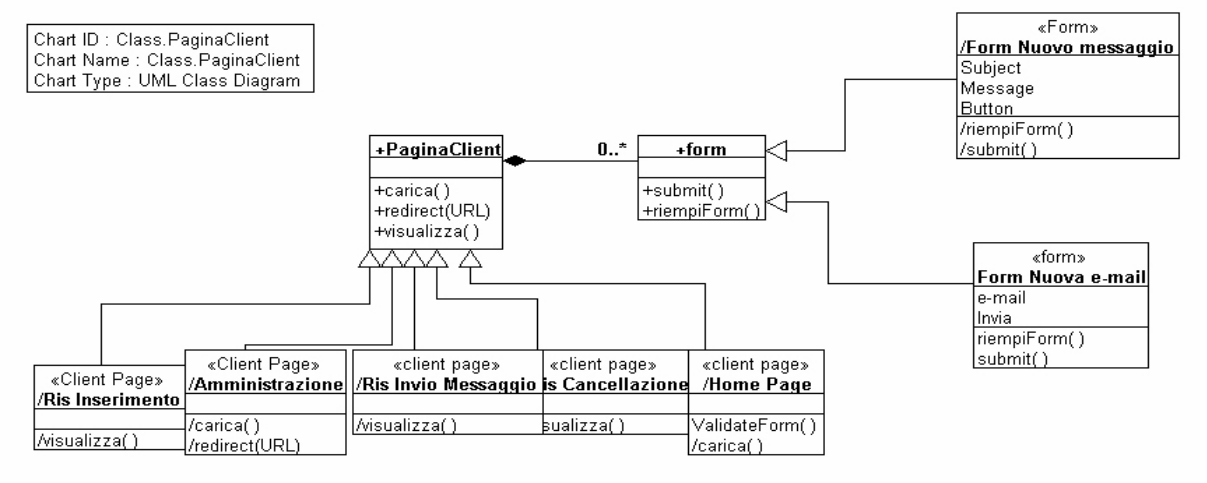


Figura 15 Gerarchia delle pagine client

2.6 Class diagram dell'applicazione mailing list

A partire dai diagrammi di tipo generale, sono stati generati quelli specifici della applicazioni; questi sono riportati nel seguito con riferimento a ciascun use case individuato.

Use case "Inserimento nuovo utente"

Il class diagram deriva dalle considerazioni fatte in sede di individuazione delle classi. Il diagram, a differenza dei diagrammi dei collegamenti, presenta anche le classi con cui le pagine server si interfaccciano e i relativi metodi.

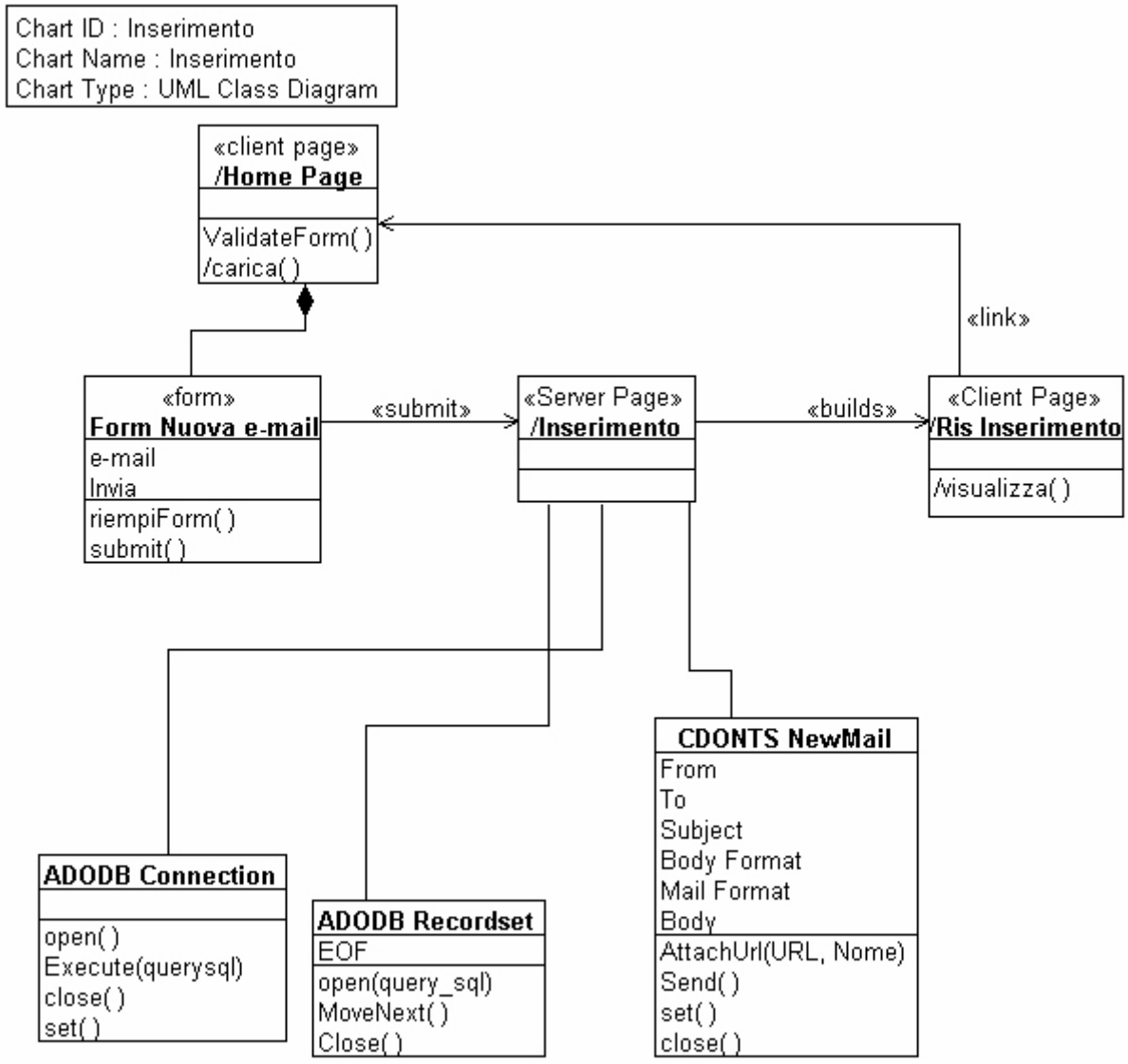


Figura 16 Class diagram dell'use case Inserimento

Use case cancellazione

Il class diagram descrive ciò che è stato anticipato in sede di individuazione delle classi.

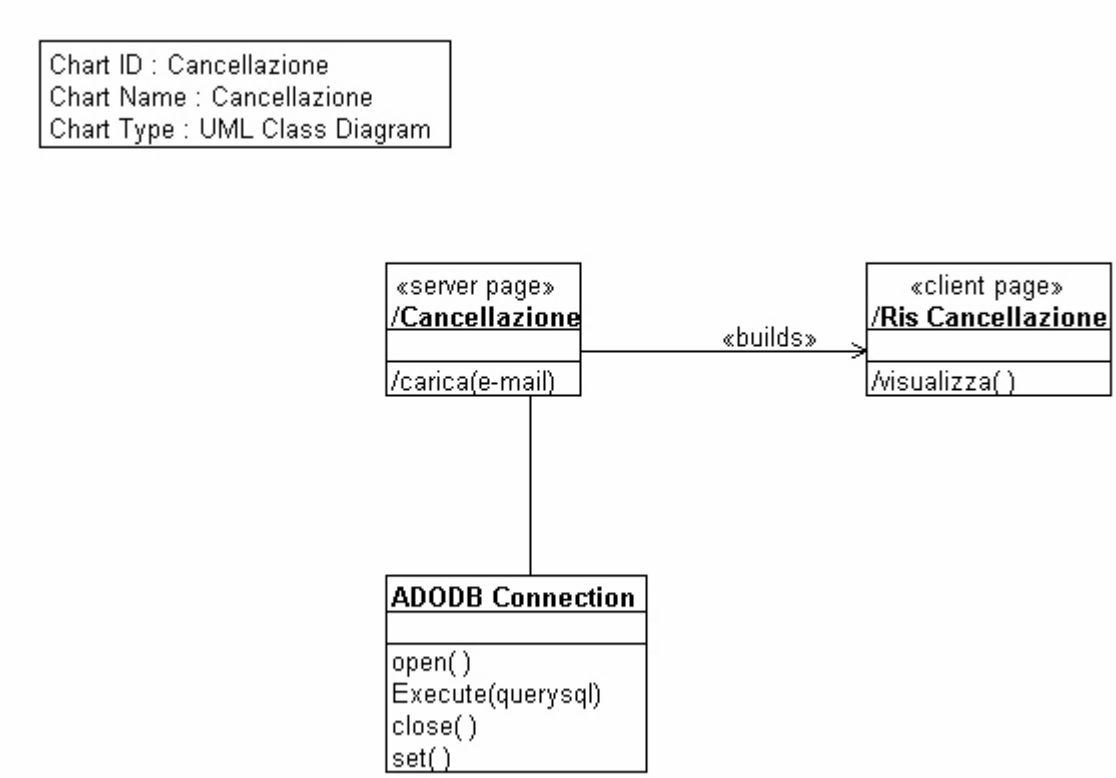


Figura 17 Class diagram dell'use case Cancellazione

Use case "Invio messaggio"

Il class diagram riflette ciò che è stato individuato con il diagramma dei collegamenti.

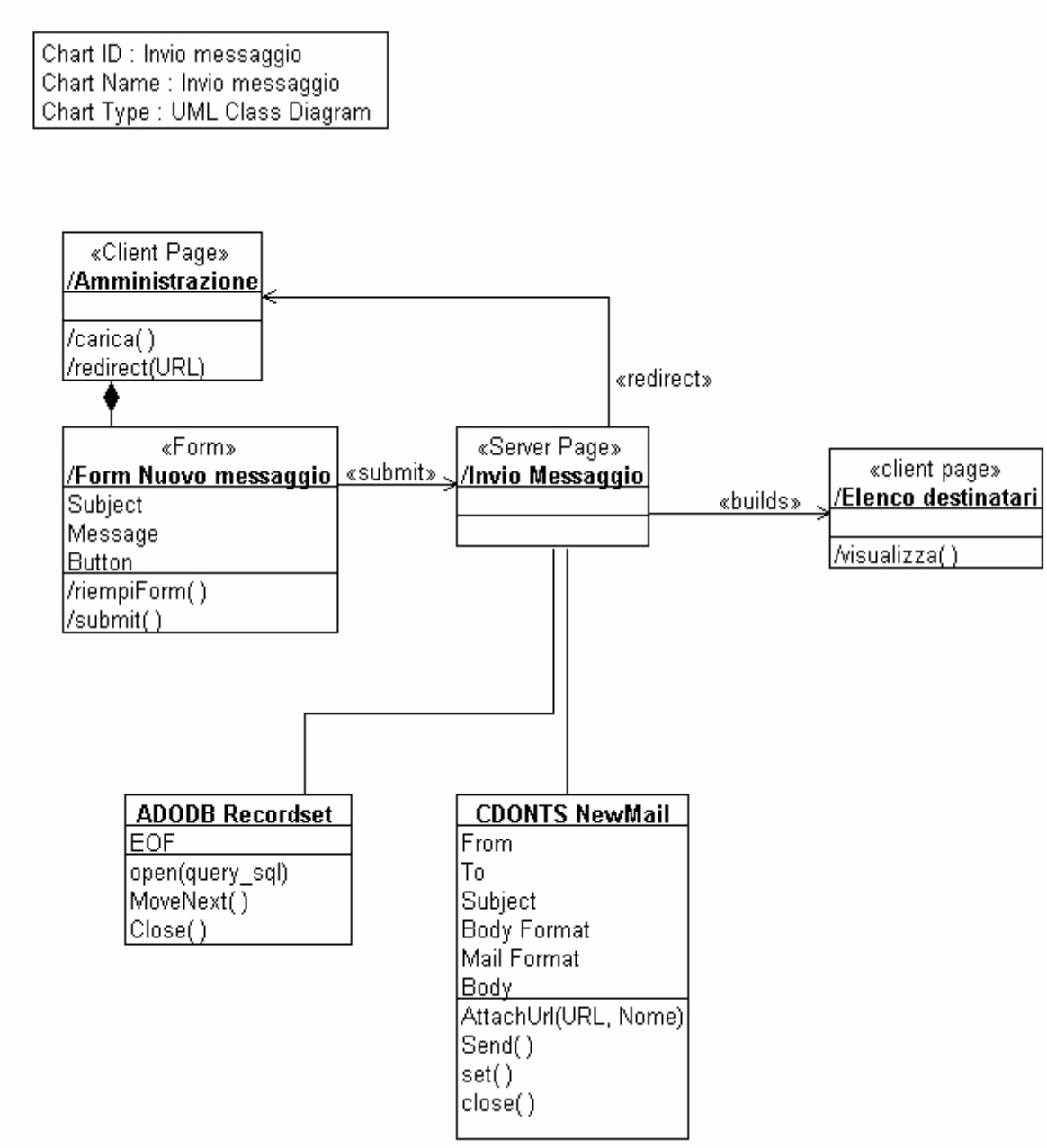


Figura 18 Class diagram dell'use case Invio messaggio

2.7 Sequence diagram

In questa sezione si vuole porre l'attenzione sulla dinamica dell'interazione tra l'utente e il sistema composto dalle classi individuate. Per ogni use case viene presentato un sequence diagram che descrive tale interazione.

Use case "Inserimento nuovo utente"

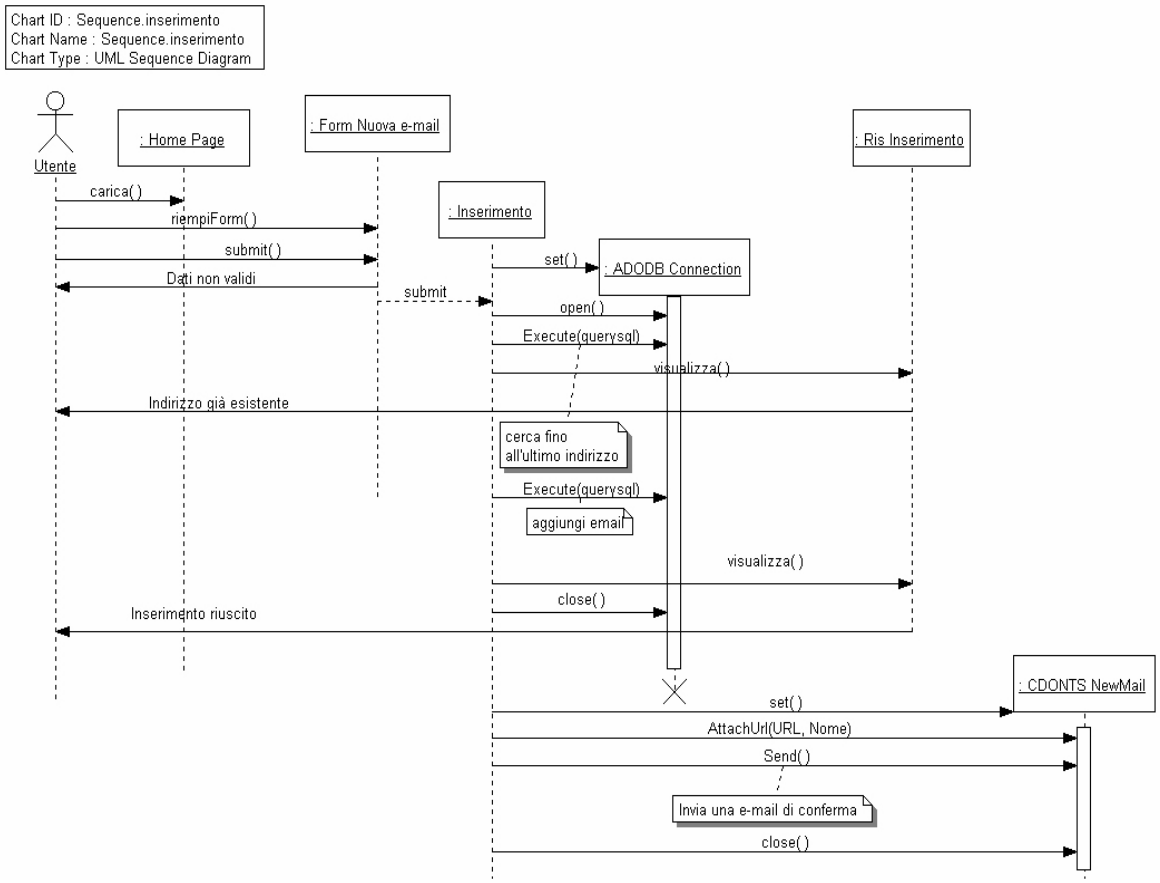


Figura 19 Class diagram dell'use case Inserimento

Use case "cancellazione"

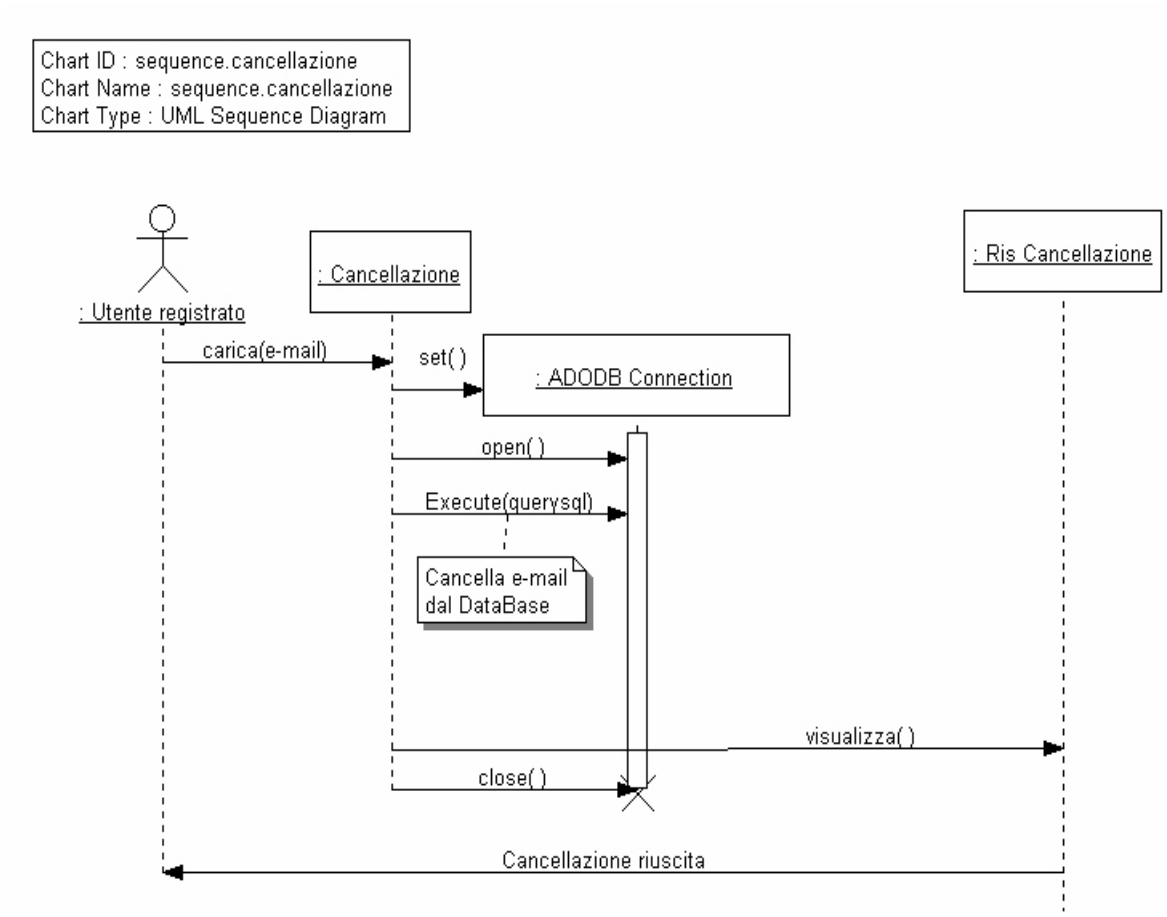


Figura 20 Class diagram dell'use case cancellazione

Use case "invio messaggio"

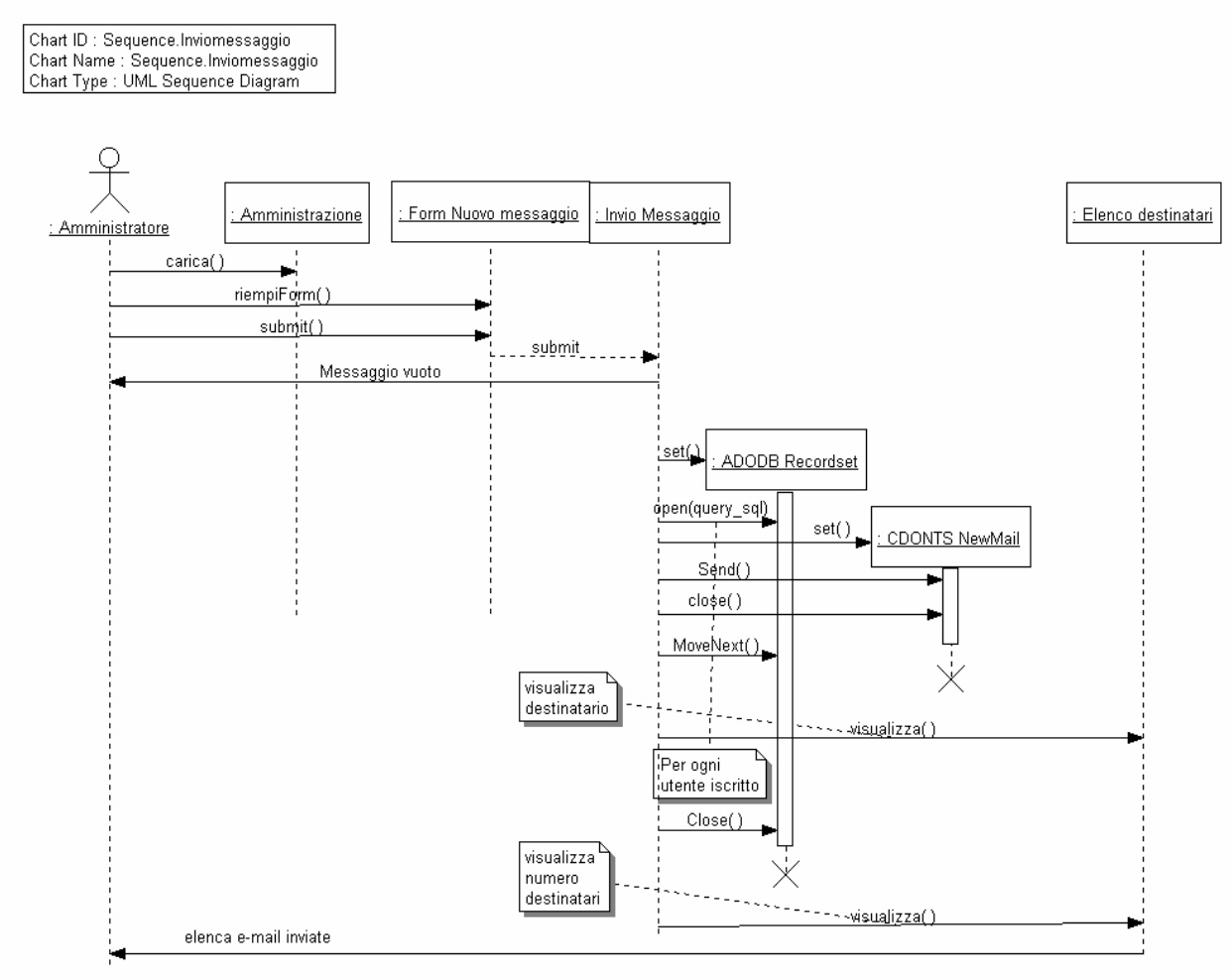


Figura 21 Class diagram dell'use case Invio messaggio

2.8 Component diagram

In tale diagramma è indicato quali componenti fisici realizzeranno le varie classi. I componenti fisici sono tipicamente dei file e , quindi, l'organizzazione si basa sul principio che in genere una pagina client è contenuta in un file HTML con estensione . htm, una pagina server è contenuta in un file con estensione .asp (estensione predefinita della tecnologia ASP) e una pagina client costruita da una pagina server è contenuta nello stesso file della pagina server.

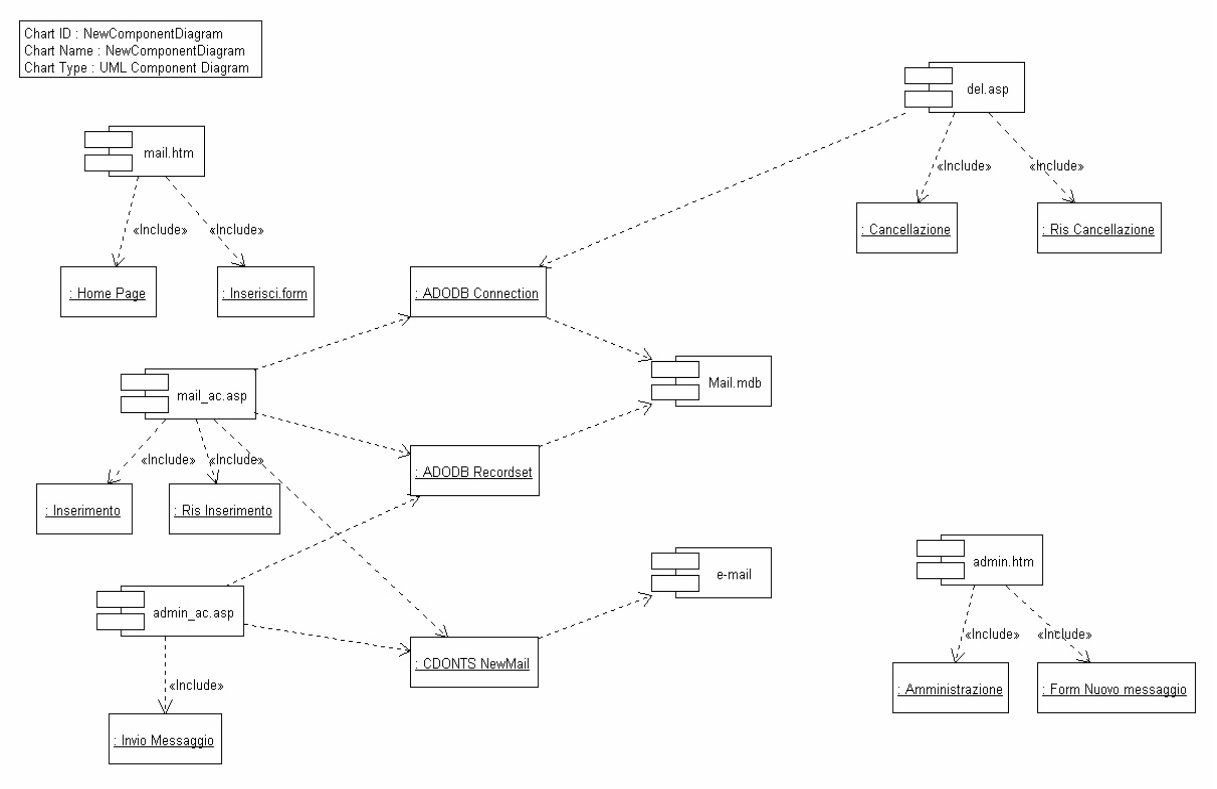


Figura 22 Component diagram

Oltre a tali file bisogna prevedere la presenza di un database che contenga le news inserite e i dettagli. Nella figura 22 si evidenzia come l'accesso al database avviene solo mediante gli oggetti di tipo ADODB.

2.9 Implementazione

Alcune scelte implementative sono già state prese e qui le riassumiamo:

- Scelta della tecnologia ASP per le pagine dinamiche dal lato server
- Scelta di Microsoft Access per realizzare il database

Viene scelto il linguaggio HTML per realizzare le pagine client e VBScript come linguaggio di scripting per realizzare le pagine .asp.

Per tale tipologia di applicazioni, si possono fare varie scelte dell'interfaccia grafica, senza che queste scelte influenzino il progetto già realizzato. Per tale motivo non ci soffermiamo sul contenuto dei singoli file da implementare, anche perché i componenti più significativi sono stati fissati in fase progettuale.

CAPITOLO 5 - REVERSE ENGINEERING DI UN'APPLICAZIONE WEB

1 Il Reverse Engineering

L'informatizzazione ha già raggiunto molti settori della società, automatizzandone processi e procedure. Nel contempo, l'informatica ha continuato la sua evoluzione in maniera velocissima, rendendo continuamente obsoleti software e sistemi in uso. In questa situazione un problema che si viene a porre sempre più spesso è quello di rinnovare i vecchi sistemi informatici, cercando però di non stravolgerne le basi che sono ancora valide. Per poter operare in questa direzione è sicuramente necessaria la comprensione del cosiddetto software ereditato, al fine di poterne ricavare informazioni relative alla sua struttura e alla sua documentazione, specie nei casi in cui questa è carente.

In questo quadro si colloca il Reverse Engineering (RE) che può essere definito come [24]:

"Il Reverse engineering è un insieme di teorie, metodologie e tecnologie per:

- Il progetto e l'implementazione di processi di estrazione ed astrazione di informazioni da componenti di un sistema software esistente per produrre nuovi componenti ad un livello di astrazione maggiore e consistenti con quelli utilizzati;
- L'aggiunta ai componenti prodotti di conoscenza ed esperienza che non può essere ricostruita direttamente ed automaticamente dai componenti analizzati"

Il reverse engineering parte, quindi, dall'analisi di oggetti di un sistema software e ha come obiettivo la produzione di oggetti del sistema in esame ad un livello maggiore di astrazione, che possa fornire una descrizione dei componenti del sistema in esame e delle relazioni tra essi. Non è quindi associato al reverse engineering nessun cambiamento del codice sorgente che viene visto come l'input di tutto il processo. I risultati del processo di reverse engineering possono variare da diagrammi che visualizzano la struttura del sistema in esame a documenti che descrivono le funzionalità implementate dal sistema. Con riferimento ai diagrammi UML si possono ricavare per RE ad esempio i class diagram, gli use case diagram, i sequence e collaboration diagram, i component diagram.

Il reverse engineering si colloca nel ciclo di vita del software a supporto delle attività di manutenzione, ma può trovare applicazione in ogni caso in cui è necessario ricavare i componenti di un sistema software e le relazioni tra essi.

Quando si richiede un intervento di manutenzione su un software di cui non sono disponibili i necessari documenti, la prima attività consiste nel ricostruire i modelli del software mediante un processo di reverse engineering.

Il processo di reverse engineering si divide in due fasi principali:

- Fase di estrazione, nella quale si cerca di ricavare dalle risorse sorgenti (spesso solo il codice) quante più informazioni possibili. Tale fase consiste tipicamente in un'analisi statica del codice sorgente, in genere effettuata mediante uno strumento automatico (estrattore o analizzatore). Le informazioni ricavate in questa fase possono già costituire una base per ricavare dei modelli associati alla struttura dei singoli file sorgenti analizzati. Molti degli analizzatori in uso memorizzano le informazioni ricavate dai file sorgenti in un repository che può quindi essere considerata come un primo modello di astrazione del sistema software in esame;
- Fase di astrazione, nella quale si cerca di dare un significato alle informazioni ricavate costruendo i modelli finali mediante strumenti software detti astrattori, che elaborano le informazioni ottenute dalla fase di estrazione e forniscono una serie di modelli del software in esame che sono appunto l'output di tutto il processo. Tali strumenti in genere consentono anche l'interazione all'utente che collabora alla definizione degli output.

Ogni processo di reverse engineering è associato a particolari modelli di output e organizzativi del processo stesso.

Un paradigma utilizzato per modellare il processo di reverse engineering, è il paradigma Goals/Models/Tools. Tale paradigma dovrebbe essere a supporto della fase di progettazione del processo di RE, e quindi fornire una definizione dei requisiti degli estrattori e degli astrattori. Vediamo ora brevemente le caratteristiche salienti delle tre fasi:

- Goals: si tratta della fase in cui cercare di individuare quali possano essere le viste astratte del sistema software in esame;
- Models: in questa fase si vogliono individuare esattamente i modelli di rappresentazione delle informazioni estratte e quindi identificare tutti i requisiti necessari alla loro compilazione ;
- Tools: in questa fase invece si punta direttamente verso gli estrattori e gli astrattori in quanto si vogliono definirne precisamente i requisiti nonché le scelte progettuali.

Il reverse engineering in generale può applicarsi sicuramente anche alle applicazioni web. Nell'ambito delle applicazioni web , infatti, si sente l'esigenza di costruire modelli per descriverne i componenti e le relazioni tra essi, visto che tipicamente per le applicazioni web non è seguito un definito processo di sviluppo e quindi esse non sono accompagnate da una adeguata documentazione. Tali applicazioni, infatti, difficilmente nascono da una lunga fase di progetto ma, più spesso, da una frettolosa implementazione, a causa delle pressanti richieste del mercato.
In questa situazione le applicazioni web tendono velocemente a diventare delle giungle nelle quali è molto difficile riconoscere il progetto iniziale, in modo da poterlo manutenere o espandere. Possono, quindi, essere di enorme aiuto strumenti automatici che svolgano almeno una parte del processo di reverse engineering di applicazioni web.

Al fine di poter costruire alcuni strumenti a supporto del reverse engineering di applicazioni web, può essere necessario uno studio preliminare di alcune semplici applicazioni web per poter meglio individuare e definire i goals e i models del processo di reverse engineering . Nei prossimi paragrafi è descritto il reverse engineering di un'applicazione web, effettuatocome esperimento per definire un processo di reverse engineering nonché i modelli di astrazione ricavabili per tali applicazioni.

2 Reverse engineering di un'applicazione web

E' stata scelta un'applicazione per gestire una mailing list client-server di cui si hanno a disposizione i soli file sorgenti e nessuna altra documentazione.
L'applicazione scelta è semplice in modo da poter effettuare "a mano" e senza troppo sforzo il suo RE, ma presenta tutti quegli aspetti caratterizzanti un'applicazione web.
Le astrazioni e i modelli che si vogliono ottenere sono quelli precedentemente usati in forward engineering.

2.1 Analisi del codice sorgente

Innanzitutto è stato effettuato un "inventario" dell'applicazione per individuare tutti i file di codice sorgente che la implementano.
Il codice sorgente è risultato essere contenuto nei seguenti files:

- addnews.asp
- addnews.html
- delete.asp
- delete.html
- elenco.asp
- index.asp
- news.asp
- news.mdb

Per ottimizzare l'analisi è possibile stabilire un ordine con il quale analizzare i file. Come metodologia generale, si può scegliere di incominciare dai file che rappresentano pagine web, in modo da riconoscere subito il percorso di navigazione. Inoltre, ipotizzando una semantica significativa dei nomi dei file, essi sono stati raggruppati in cinque gruppi: nel primo sono stati inseriti i file addnews.*, nel secondo delete.*, nel terzo elenco.asp, nel quarto index.asp, infine nel quinto news.asp e sono stati analizzati in quest'ordine, cominciando dalle pagine passive .htm, procedendo poi con le pagine attive .asp, concludendo con gli altri file.
Le motivazioni di questa scelta sono state puramente euristiche: dal momento che non vi erano subdirectory, si é pensato che era il solo nome del file a fornire indicazioni sulla sua funzione, per cui nomi simili dovrebbero essere utilizzati per denominare file che compartecipano ad una stessa funzionalità.

Per quanto riguarda invece l'ordine con il quale esaminare le pagine all'interno di un sottogruppo, la strategia è consistita nel partire dalle pagine con le quali dalle quali l'utente inizia la navigazione. In mancanza di ulteriori informazioni, è più probabile che la navigazione incominci da una pagina passiva. Ciò perché le pagine attive sfruttano meglio le proprie potenzialità quando sono richiamate in maniera parametrica. Ciò non toglie che in una sola pagina asp potrebbero anche essere racchiuse tutte le funzionalità di un intera applicazione web.
In considerazione di quelli che sono i modelli usati nell'esempio precedente di forward engineering, l'obiettivo prefisso é stato quello di concentrare l'attenzione su tutti quegli aspetti legati all'interazione tra gli oggetti, cercando di esulare da quelli legati alla visualizzazione.
Più precisamente, si è deciso di ricavare i seguenti diagrammi: use case di alto livello, diagrammi dei collegamenti, class diagram e sequence diagram.
Ovviamente non si è preteso, con questo primo esperimento, di ottenere risultati pienamente soddisfacenti, ma soprattutto di poter individuare e definire una metodologia e cercare di comprendere i requisiti di eventuali tool automatici di supporto.

2.2 Analisi statica dei file costituenti l'applicazione

L'analisi statica dei file sorgenti, così come le altre attività, è stata effettuata "a mano". Man mano che si sono incontrati elementi caratterizzanti l'applicazione, questi sono stati annotati per poi riportarli nei diagrammi. Nel seguito è descritta l'analisi effettuata per ciascun file sorgente.

File addnews.html

Si tratta di una pagina client html e verrà indicata come tale nel diagramma dei collegamenti. Viceversa, nel class diagram essa sarà rappresentata da una classe con lo stereotipo Pagina Client. Ciò vale per ogni file con questa estensione.
Il primo elemento rilevante che si incontra scorrendo il codice è un form (metodo post) che indirizza verso *addnews.asp*. I parametri di questo form sono i seguenti:

- autore, di tipo text
- email, di tipo text
- titolo, di tipo text

- fonte, di tipo text
- link, di tipo text
- news, di tipo textarea
- encode, di tipo radio

Ad essi si aggiungono i due parametri standard di una form di input, ovvero Reimposta (di tipo reset) e Spedisci (di tipo submit). Nel class diagram si deve inserire una classe form, prodotta dalla classe Pagina Client, elencando tutti i parametri, nonché il metodo submit. Nel diagramma dei collegamenti si inserirà il form e l'arco submit che raggiunge la Pagina Server *addnews.asp* che ora si andrà ad esaminare.

File addnews.asp

In questo caso si ha a che fare con una pagina asp, in cui è possibile avere blocchi scritti in tale linguaggio frammisti a porzioni di codice html che andranno a formare una pagina client che si etichetterà con lo stereotipo pagina client costruita e con lo stesso nome della pagina server. Nel diagramma dei collegamenti si inizia per ora a inserire una pagina server, mentre nel class diagram si inserisce una classe con stereotipo pagina server.

Si trova subito un blocco VBScript, ma senza informazioni rilevanti, poi un commento, poi un altro blocco VBScript.

In questo secondo blocco si leggono i dati provenienti dal form, con il metodo predefinito *request.form,* usando talvolta anche il metodo *Server.HTMLEncode* per l'eventuale decodifica.

E' poi creata la connessione fisica con il database, utilizzando il metodo predefinito *Server.CreateObject* e poi aperta con il metodo *open*.

Per poter operare sul database viene creato l'oggeto RecordSet (ancora con il metodo predefinito *Server.CreateObject*) e aperto con il metodo open.

Sull'oggetto recordset si impostano i valori dei campi, utilizzando i metodi *addnew* e *fields*, infine si aggiorna il database con il valore del recordset costruito, con il metodo *update*. A questo punto vengono chiusi prima il recordset, poi il database (sempre con il metodo predefinito *close*).

Tutte queste interazioni con il database e con l'oggetto recordset sono descritte nel class diagram, come riferimento a metodi predefiniti, ma serviranno soprattutto alla compilazione del sequence diagram, per il quale si rimanda all'analisi successiva.

Il file termina con l'utilizzo del metodo *response.redirect* il quale serve a ridirigere il controllo verso *index.asp*. Quest'interazione verrà segnalata sia nel diagramma dei collegamenti che nel class diagram.

A questo punto si può fare l'ipotesi che i due file appena visti realizzino un unico use case, in quanto si è visto come non esistano legami con altre pagine web dell' applicazione. Si tracciamo quindi i diagrammi, che compaiono nelle figure seguenti.

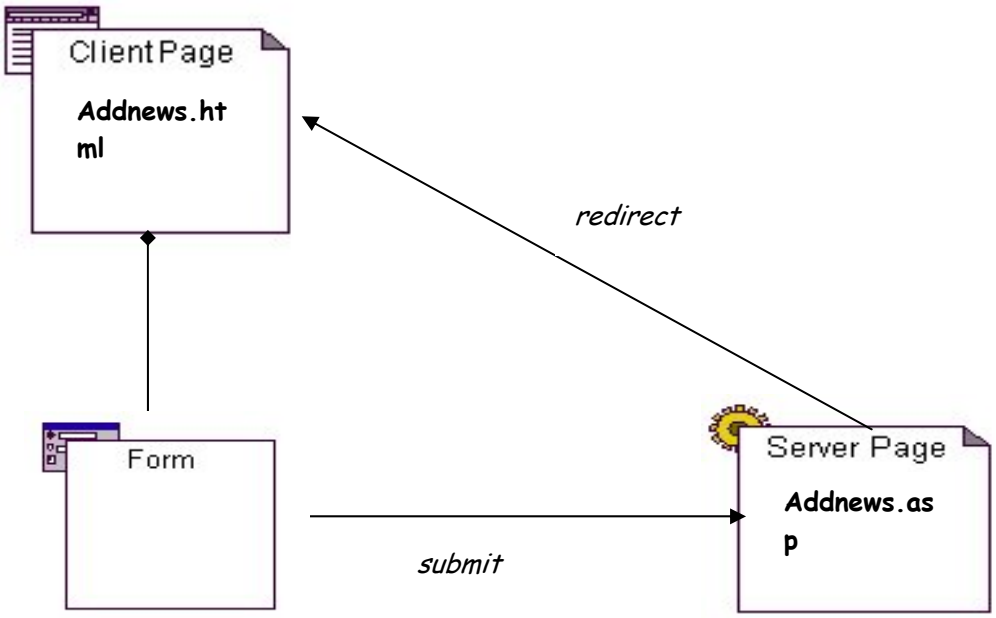


Figura 1 - Diagramma dei collegamenti tra le pagine addnews.html e addnews.asp

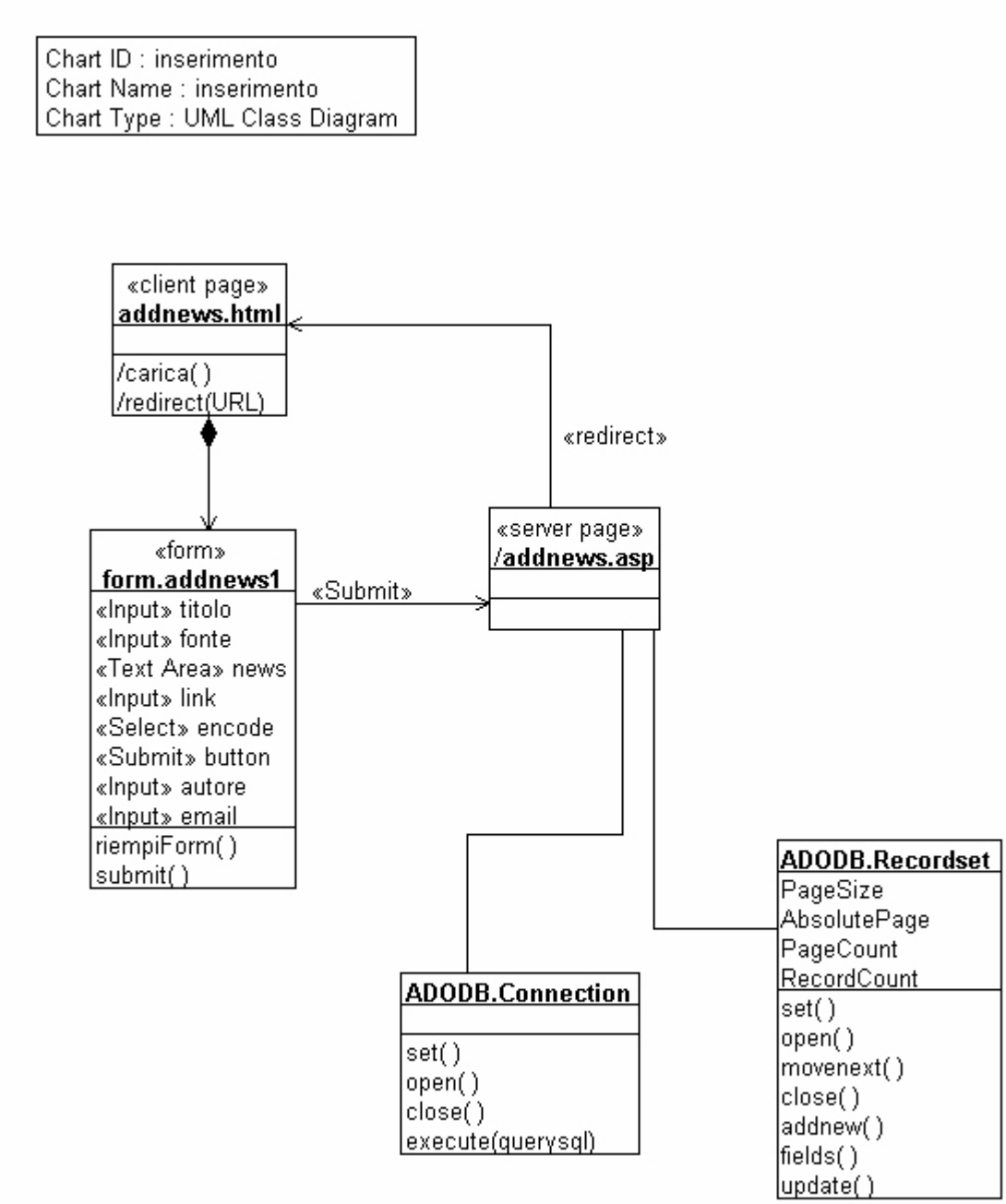


Figura 2 - Class diagram risultante dall'analisi statica dei file addnews.html e addnews.asp

File delete.html

Si tratta di una Pagina Client, quindi si inserisce nei diagrammi come già visto per *addnews.html*.

Scorrendo il codice si incontra un form (metodo post) che punta verso la pagina server *delete.asp*.

I primi tre parametri del form che si incontrano sono autore, email e codec. Tutti e tre questi parametri sono di tipo hidden, ovvero non possono essere modificati dall'utente, ed hanno dei valori predefiniti. Questo meccanismo è utilizzato per poter inviare dei parametri alla pagina server, senza doverli includere nella riga di comando.

C'è poi il parametro sim, di tipo radio, che può assumere tre valori possibili e il parametro newsid, di tipo text. Infine ci sono i due parametri tipici delle form di input, Reimposta di tipo reset e Spedisci di tipo submit. A questo punto termina la definizione del form ed anche il file.

File delete.asp

Il codice comincia con l'apertura di un blocco VBScript, nel quale si ricavano i valori dei parametri del form, ancora con il metodo *request.form*.

In un ramo else di un select, si trova anche il metodo predefinito *response.redirect* il cui parametro assume il valore della variabile *sito*. In questo caso non è possibile inserire una relazione statica tra pagine web, al massimo si potrà inserire una relazione verso una pagina web non specificata. Non si include quest'interazione nel diagramma dei collegamenti, mentre segnaliamo la chiamata del metodo redirect nel class diagram.

Nel successivo blocco VBScript c'è l'utilizzo del il database con la creazione di un'interfaccia (metodo *Server.CreateObject*), l'apertura (metodo *open*), l'esecuzione di una query (metodo *execute*) e la chiusura dell'interfaccia (metodo *close*).

Il file termina con un altro redirect, identico al precedente.

A questo punto, siccome non ci sono interazioni statiche con altre pagine, si possono tracciare il diagramma dei collegamenti e il class diagram relativo al gruppo composto dai due file *delete.asp* e *delete.html*.

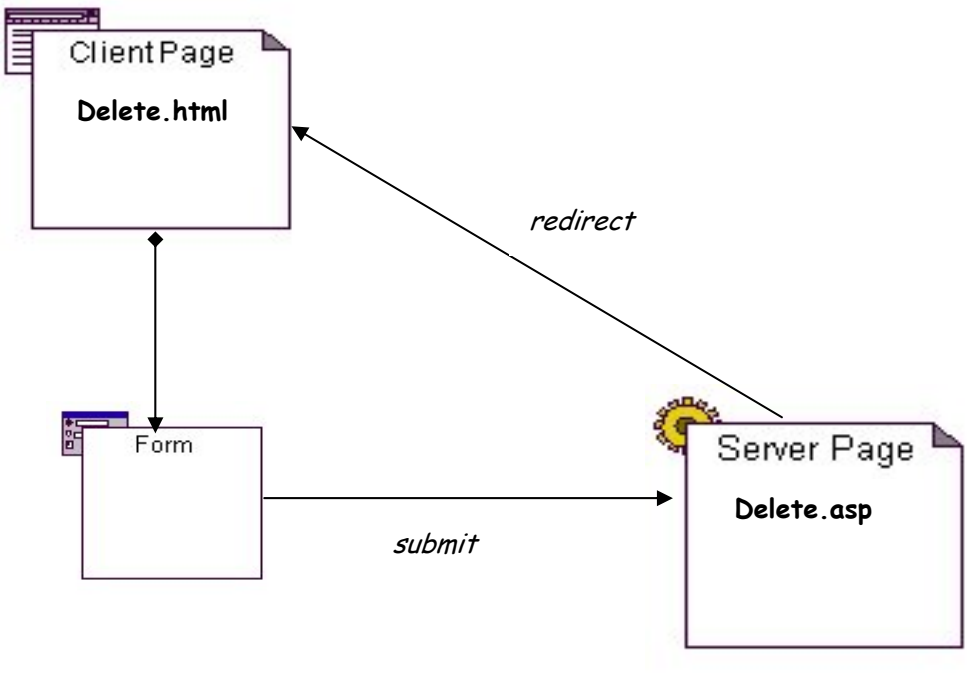


Figura 3 - Diagramma dei collegamenti tra le pagine delete.html e delete.asp

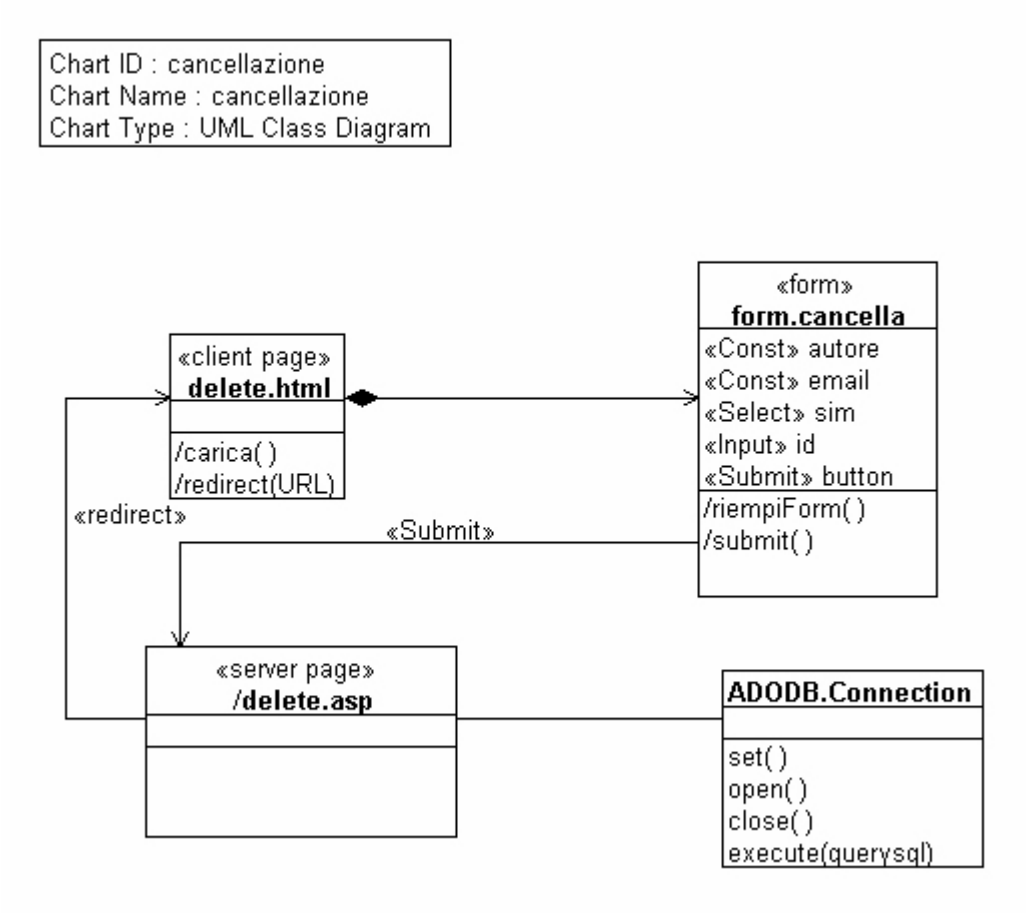


Figura 4 - Class diagram risultante dall'analisi statica dei file delete.html e delete.asp

File elenco.asp

Scorrendo il codice si incontra prima un blocco VBScript senza metodi rilevanti, poi del codice html comprendente anche dei tag di visualizzazione. In questo caso quindi siamo sicuri che la pagina asp fornirà un output al browser sotto forma di pagina client costruita. Nel class diagram bisogna inserire un'ulteriore classe con lo stereotipo pagina client costruita. L'associazione tra le due classi avrà lo stereotipo builds. Aggiunte analoghe verrannno fatte al diagramma dei collegamenti.

Successivamente c'è un blocco VBScript senza metodi rilevanti, poi un altro blocco VBScript nel quale si ricava un parametro, stavolta dalla riga di comando con il metodo predefinito *Request.QueryString*. A questo punto vengono prima create e poi aperte una connessione al database e una ad un recordset, utilizzando i metodi già visti per *addnews.asp.*

Nell'elaborazione dell'oggetto recordset vengono utilizzati alcuni metodi predefiniti: *rs.PageSize, rs.AbsolutePage, rs.EOF, rs.movenext.*

C'è inoltre un punto molto rilevante: in una zona di codice html, c'è un'ancora verso la pagina *news.asp* ma con un parametro sulla linea di comando che, anziché essere una stringa costante, è costituito da una prima parte costante e da una seconda parte che assume il valore di una variabile VBScript (a questo scopo viene aperto e poi immediatamente richiuso un blocco VBScript). In questo caso possiamo considerare a tutti gli effetti quest'ancora come diretta alla suddetta pagina, ma dobbiamo tener presente che la semantica di quest'interazione sarà decisa solo a tempo di esecuzione.

Continuando a scorrere il codice troviamo un'altra situazione complessa: all'interno di un blocco VBScript, nel ramo then di un if c'è un'ancora html: significa che quel link esisterà a tempo di esecuzione solo se sarà verificata la condizione dell'if. In fase statica consideriamo comunque come esistente tale interazione. Questa situazione si ripete in maniera identica anche una seconda volta, dopodichè si trovano la chiusura del recordset e dell'interfaccia con il database (sempre con il metodo *close*).

A questo punto non possiamo considerare concluso il sottogruppo contenente la sola pagina elenco.asp, in quanto essa comunica anche con *news.asp*. Consideriamo quindi anche quest'ultima nel gruppo e ne anticipiamo l'esame.

File news.asp

L'analisi di questa pagina server porta, per prima cosa, ad individuare l'utilizzo del metodo predefinito *request.QueryString* che permette di ricavare il valore del parametro passato sulla linea di comando. Successivamente vengono creati ed aperti il database e un recordset, con i soliti metodi. Sul recordset viene questa volta applicato il metodo predefinito *RecordCount*.

A questo punto troviamo l'utilizzo del metodo predefinito *Response.Redirect* verso la pagina *index.asp*. Oltre ad aggiungere al solito modo quest'informazione nel diagramma dei collegamenti e nel class diagram, si può anche concludere che index.asp appartiene anch'essa al gruppo in esame.

Dopo questo blocco VBScript si trova una serie di tag HTML che contribuiscono anche alla visualizzazione. Si procede quindi all'inserimento in entrambi i diagrammi di una Pagina Client Costruita come visto per *elenco.asp*.

Anche in quest'occasione si ritrova un'ancora a *news.asp* che ha un valore sulla riga di comando che dipende da variabili VBScript. Infine, in un ulteriore blocco VBScript, vengono chiusi il recordset e la connessione al database.

File index.asp

Questa pagina server comincia con dei tag html, alcuni dei quali riguardanti la visualizzazione: viene quindi inserita una pagina client costruita con le modalità più volte viste.

In un blocco Vbscript si incontrano i metodi che permettono la creazione e l'apertura dell'interfaccia con il database e di un recordset. Su questo recordset viene poi eseguito anche il metodo movenext e la proprietà EOF.

Anche questa volta c'è un link a *news.asp* con un parametro variabile.

Infine vengono chiusi il recordset e il database.

A questo punto si possono finalmente tracciare i diagrammi relativi a questo terzo e ultimo sottogruppo:

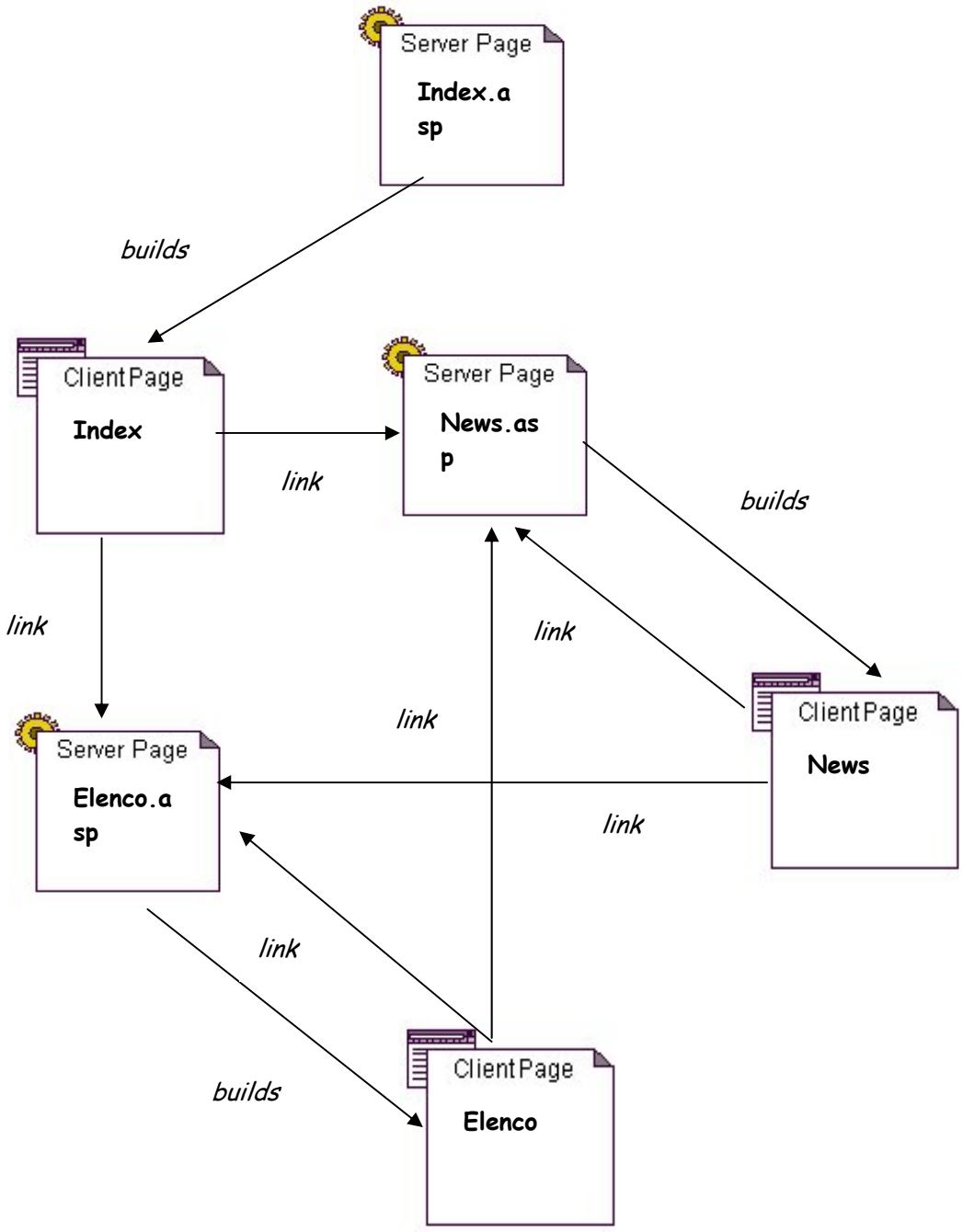


Figura 5 - Diagramma dei collegamenti tra le pagine elenco.asp, index.asp e news.asp

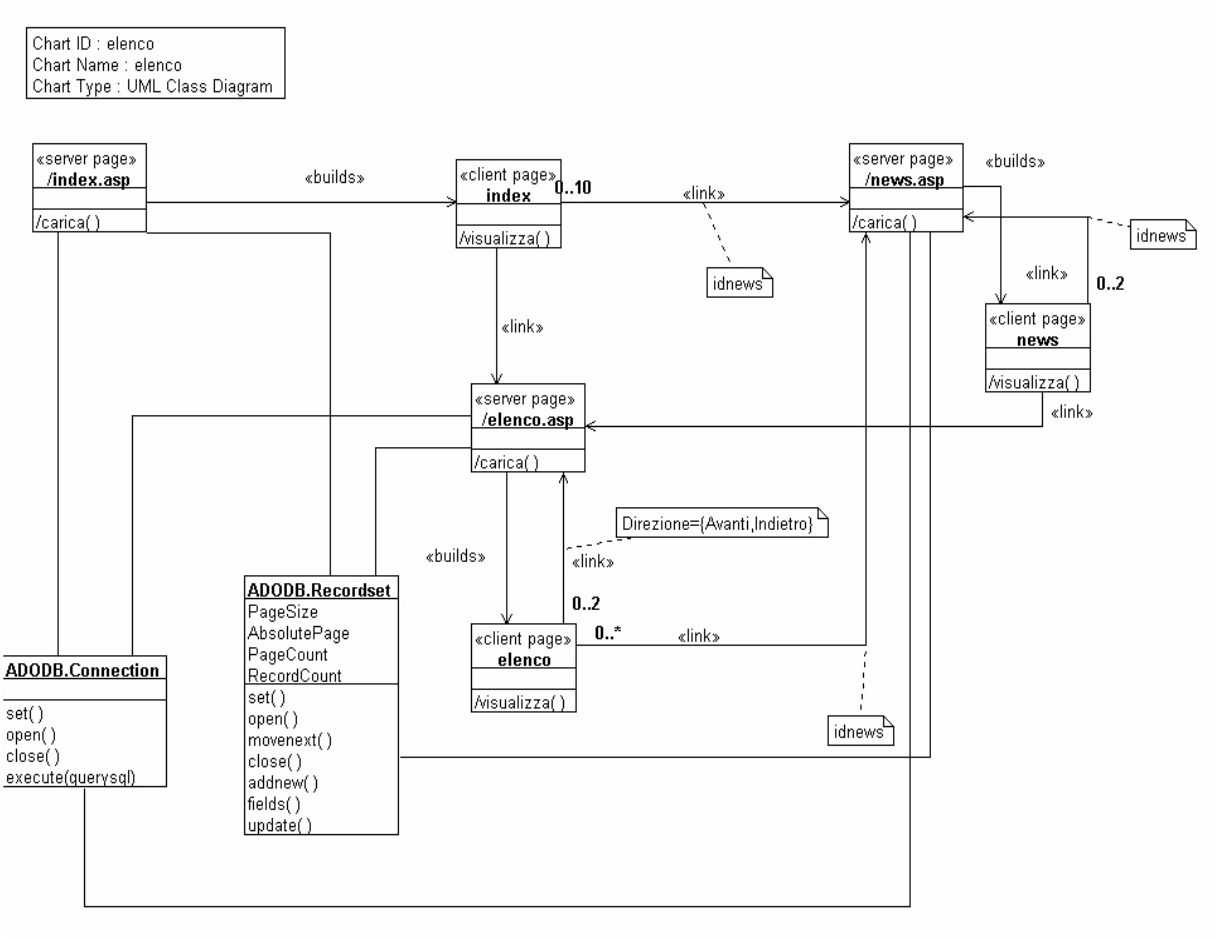


Figura 6 - Class diagram risultante dall'analisi statica dei file elenco.asp, index.asp e news.asp

Con ciò si può considerare completata la rappresentazione statica del sistema, almeno nell'ambito delle viste scelte. In realtà ci sono altre viste statiche significative, che comprendano ad esempio i collegamenti agli altri oggetti, come ad esempio le immagini oppure i database. Anche così non si può però raggiungere una vista esaustiva del sistema. Infatti un'applicazione web è tutt'altro che una struttura chiusa, della quale si possono delimitare esattamente i confini, e ulteriori collegamenti verso oggetti diversi potrebbero venirsi a configurare solo in fase dinamica.

Dall'analisi dei file sorgenti sono state trovate alcune interazioni delle pagine server con classi predefinite di ASP. E' possibile tracciare due class diagram, uno per il lato server e l'altro per quello client, che riportino le classi predefinite utilizzate.

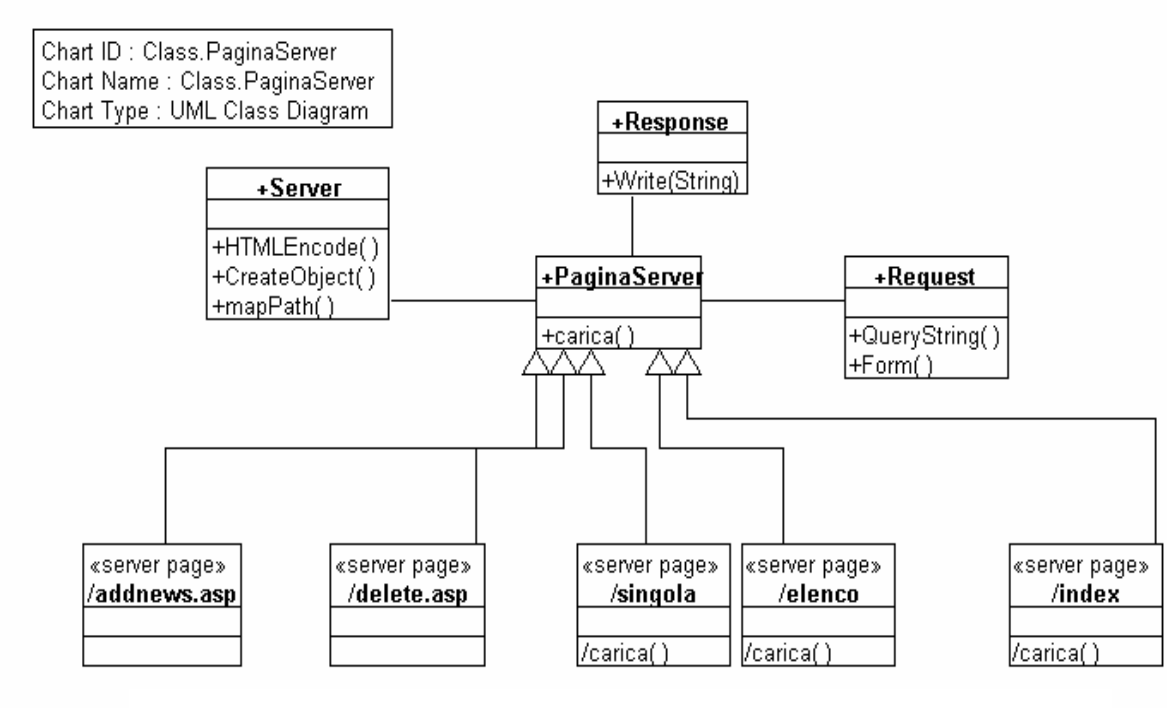


Figura 7 - Class diagramche mostra le classi predefinite del linguaggio a partire da PaginaServer, limitatamente ai metodi effettivamente utilizzati nell'esempio

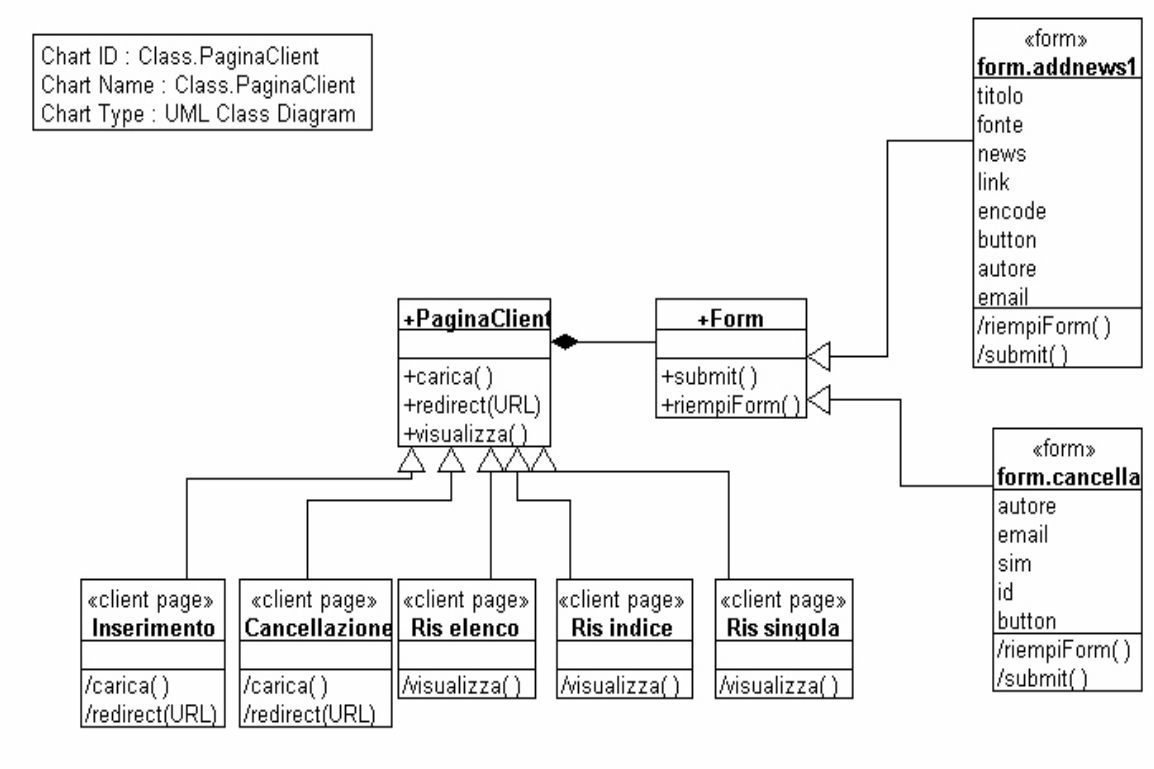


Figura 8 - Gerarchia delle pagine client

2.3 Ottenimento e redazione dei sequence diagram

I class diagram descrivono gli aspetti statici dell'applicazione in termini di suoi componenti e relazioni tra essi. Al fine di descrivere anche gli aspetti dinamici della stessa, vengono ricavati per RE i sequence diagram in modo da poter descrivere come gli oggetti identificati interagiscono e collaborano tra di loro per implementare i vari scenari con cui un utente può usare l'applicazione.

L'analisi statica effettuata precedentemente viene quindi raffinata per poter individuare in dettaglio i messaggi che gli oggetti si scambiano.

Nel seguito sono descritti i passi effettuati per tale analisi, condotta seguendo gli schemi di analisi dei file descritti prima.

File addnews.html e addnews.asp

Siccome la pagina addnews.asp non fornisce alcun output ed utilizza dei parametri non provenienti da linea di comando, si può affermare sicuramente che l'utente possa interagire solo con la pagina addnews.html.

Nel sequence diagram si conviene di denominare tale interazione *carica.* In generale essa esprime l'evento col quale l'utente decide, avendo avviato il proprio programma browser, di digitare l'indirizzo della pagina web in questione. La naturale risposta del browser sarà la restituzione della pagina, interpretata secondo il linguaggio HTML (se si tratta di una pagina server ci sarà una prima interpretazione del linguaggio asp ad opera del web server, poi l'interpretazione HTML del browser).

L'unico elemento interattivo della pagina addnews.html é costituito dal form di input: sarà questo inevitabilmente il successivo elemento da inserire nel sequence diagram. Siccome si é considerato il form come classe, se ne inserisce un'istanza nel diagramma. Si denomina convenzionalmente l'interazione tra utente e form come *riempiForm*. Sapendo inoltre che un utente può cambiare a piacimento i valori di un form, almeno fino a che non preme il pulsante di submit, si considera separatamente quest'altro evento denominandolo con *submit*. Conseguenza dell'evento submit è il passaggio dei parametri e del controllo alla pagina server, *addnews.asp*, che a questo punto comparirà per la prima volta nel sequence diagram.

Si analizza a questo punto il codice di *addnews.asp*. Il primo evento rilevante é la creazione dell'oggetto ADODB Connection, con la parola chiave *set*. Si tratta del primo oggetto che viene effettivamente creato, in quanto non è possibile ritrovare un

evento simile nè per le pagine, nè per il form: si pone quindi a partire da questa quota la lifeline dell'oggetto.
A questo punto viene creato un oggetto ADODB Recordset, nella stessa maniera e poi viene aperto. Su quest'oggetto vengono eseguiti i metodi addnew, fields, update.
Con il metodo close si distrugge l'oggetto recordset, per cui è in questa posizione che termina la sua lifeline nel sequence diagram. Stessa sorte capita poi all'oggetto Connection.
Infine il metodo redirect riporta il controllo verso la pagina addnews.html, che poi è quella con la quale può interagire l'utente. Tale circostanza viene restituita nel sequence diagram semplicemente con l'indicazione del metodo redirect che va dalla pagina server alla pagina client.

La maggior parte delle informazioni ricavate in questa analisi del codice sono riportate sinteticamente nel sequence diagram che segue. come si era intuito precedentemente, esso realizza effettivamente un caso d'uso del sistema, ovvero l'inserimento di una notizia da parte di un utente.

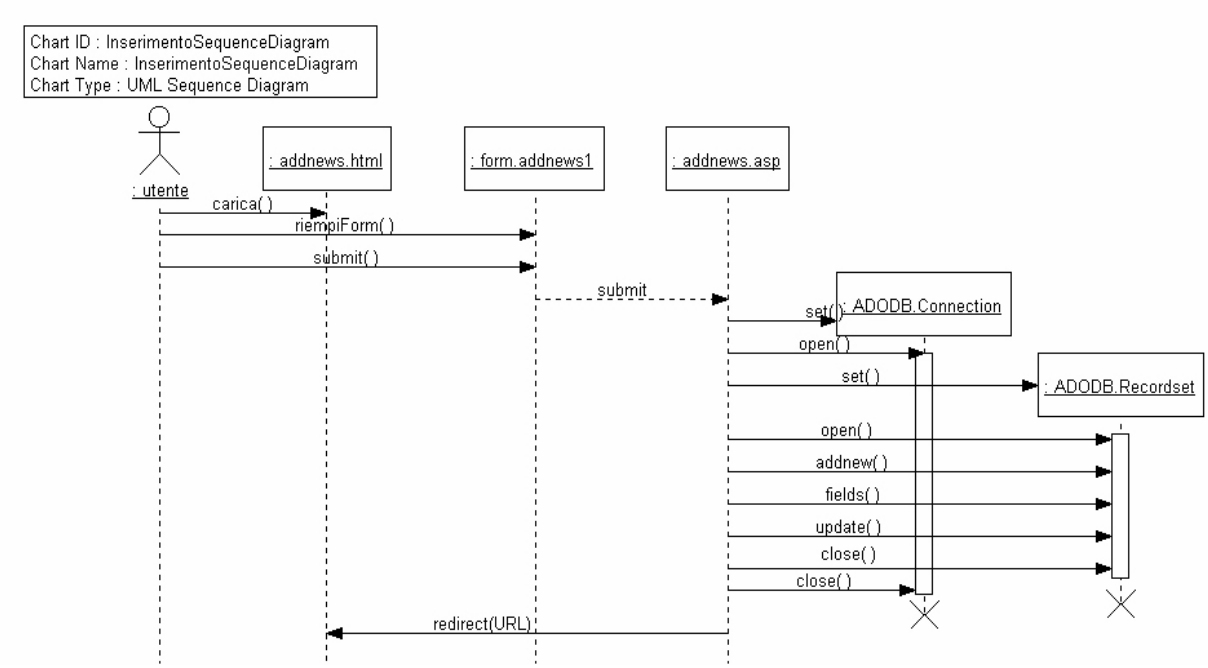


Figura 9 - Sequence diagram risultante dall'analisi dinamica del gruppo comprendente i file delete.html e delete.asp

File delete.html e delete.asp

La situazione è analoga alla precedente: un sottogruppo costituito da due file nel quale l'utente può interagire inevitabilmente solo con la pagina client, *delete.html.*

A sua volta la pagina *delete.html* è perfettamente analoga a *addnews.html*, contenendo anch'essa unicamente un form di input. Il metodo submit porta stavolta alla pagina server *delete.asp*. In questa pagina si trova ancora la creazione di un oggetto ADODB Connection, la sua apertura e l'esecuzione dei metodi execute, per avere il risultato di una query, e close, per distruggerlo.

Infine c'è il metodo redirect che, anche in questo caso, riporta il controllo verso la pagina *delete.html*.

Concludendo, questo sequence diagram è simile al precedente (l'unica differenza consiste nel mancato utilizzo di un oggetto ADODB Recordset). Una differenza sostanziale però la si può notare dall'esame dei valori dei parametri dei due form. Nel caso del form contenuto in *delete.html*, alcuni campi sono di tipo hidden, per cui non sono modificabili dall'utente e contengono appunto informazioni sull'utrnte stesso.

Si può supporre che la ragione di questa scelta derivi dal fatto che la pagina *delete.html* non sia visibile ad ogni generico utente ma solo ad una categoria particolare, che si potrebbe classificare come Amministratore, i quali sono gli unici a poter eseguire questo caso d'uso.

Non avendo trovato interazioni con altre pagine, ipotizziamo che anche questo gruppo costituisca un caso d'uso.

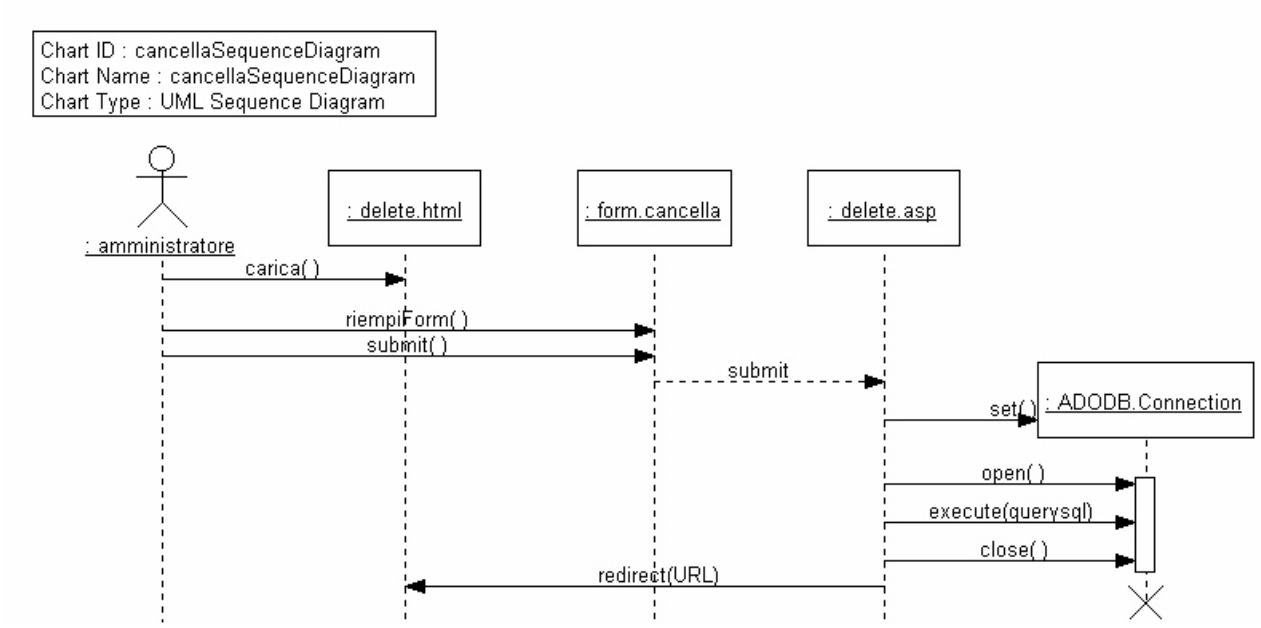


Figura 10 - Sequence diagram risultante dall'analisi dinamica del gruppo comprendente i file delete.html e delete.asp

File elenco.asp, index.asp, news.asp

Per questo gruppo la situazione è sostanzialmente diversa dai casi precedenti. Ci sono tre pagine server, ognuna delle quali però costruisce una sua pagina client, e quindi ha un'interazione con l'utente. Per decidere quale pagina sia quella da cui un utente comincia la navigazione, basta rivedere il diagramma dei collegamenti. L'unica pagina che non riceve archi entranti è index.asp, per cui è questa la pagina con cui comincerà l'interazione con l'utente.

Dopo aver supposto un'interazione fittizia (denominata ancora con carica) tra l'utente e la pagina index.asp, questa crea un oggetto ADODB Connection e un oggetto ADODB Recordset e li apre entrambi.

Successivamente, per le ultime dieci news del database, viene creata un'ancora in HTML, la quale punta a news.asp, con un parametro che varia da news a news.

Al termine, vengono chiusi e distrutti l'oggetto recordset e l'oggetto connection.

Viene inoltre mandata alla pagina client un'altra ancora, che punta alla pagina elenco.asp.

A questo punto l'esecuzione di index.asp termina, per cui si può considerare un messaggio fittizio dalla pagina client all'utente, a seguito del quale vi sia la visualizzazione dell'elenco delle ultime news. Il controllo ritorna allora all'utente, il quale dispone di ancore sia verso news.asp che verso elenco.asp.

Consideriamo per prima l'interazione con news.asp.

Quando l'utente seleziona l'ancora relativa ad una news, il controllo passa alla pagina server news.asp. Tale pagina crea e apre come al solito un oggetto ADODB connection e un oggetto ADODB Recordset. L'accesso su quest'oggetto porta alla creazione di una pagina client costruita che mostra il testo della news, nonché dei collegamenti a news.asp stesso, ma con parametro diverso, ad indicare altre news alle quali ci si può collegare.

Vengono poi chiusi i due oggetti, e viene mandato alla pagina client un'ancora ad elenco.asp. A questo punto termina l'esecuzione di news.asp, quindi bisogna inserire un altro messaggio fittizio dalla pagina client costruita all'utente, con la visualizzazione dei dettagli della news aperta.

L'utente può ora interagire in due modi con questa pagina:

- richiamare un'altra news da un'ancora a news.asp
- richiamare elenco.asp dall'ancora presente su questa pagina.

La prima possibilità ci riporta però ad un'interazione già vista, quindi non verrà riconsiderata. La seconda possibilità riguarda la stessa interazione che era rimasta arretrata, quindi verrà affrontata ora una volta per tutte.

Quando l'utente attiva la pagina server elenco.asp, essa, come al solito, crea e apre un oggetto connection e un oggetto recordset.

L'oggetto recordset è utilizzato per implementare un ciclo su ognuna delle news presenti nel database, visualizzando, nella pagina client costruita, un'ancora a news.asp con un parametro variabile in dipendenza della news considerata.

Vengono poi chiusi i due oggetti, e inviate alla pagina client altre due ancora alla pagina elenco.asp stessa, ma con parametri differenti.

Con il termine di questa pagina server, e quindi con l'invio del solito messaggio di visualizzazione dalla pagina client costruita all'utente, si può considerare terminato il sequence diagram, in quanto non sono possibili ulteriori interazioni diverse da quelle già considerate.

A questo punto però, guardando anche l'esecuzione dell'applicazione, sembra che il gruppo di file individuato realizzi tecnicamente più di un caso d'uso. Più precisamente si possono riconoscere due particolari use case, concettualmente diversi:

- visualizza news dall'elenco completo
- visualizza news dall'elenco delle più recenti.

Il primo caso d'uso si riferisce all'interazione dell'utente con elenco.asp, per la visualizzazione dell'elenco completo, e con news.asp per visualizzare una singola news.

L'altro caso d'uso si riferisce all'interazione dell'utente con l'elenco delle news più recenti, generato da index.asp, visualizzando eventualmente una di queste news.

I sequence diagram corrispondenti si ricavano quindi partizionando il sequence diagram generale e non vengono perciò riportati. Con quest'analisi non si tiene conto della possibilità di arrivare all'elenco completo a partire dall'elenco delle più recenti. In ogni caso l'attore del sistema sarà sempre lo stesso, ovvero un utente generico.

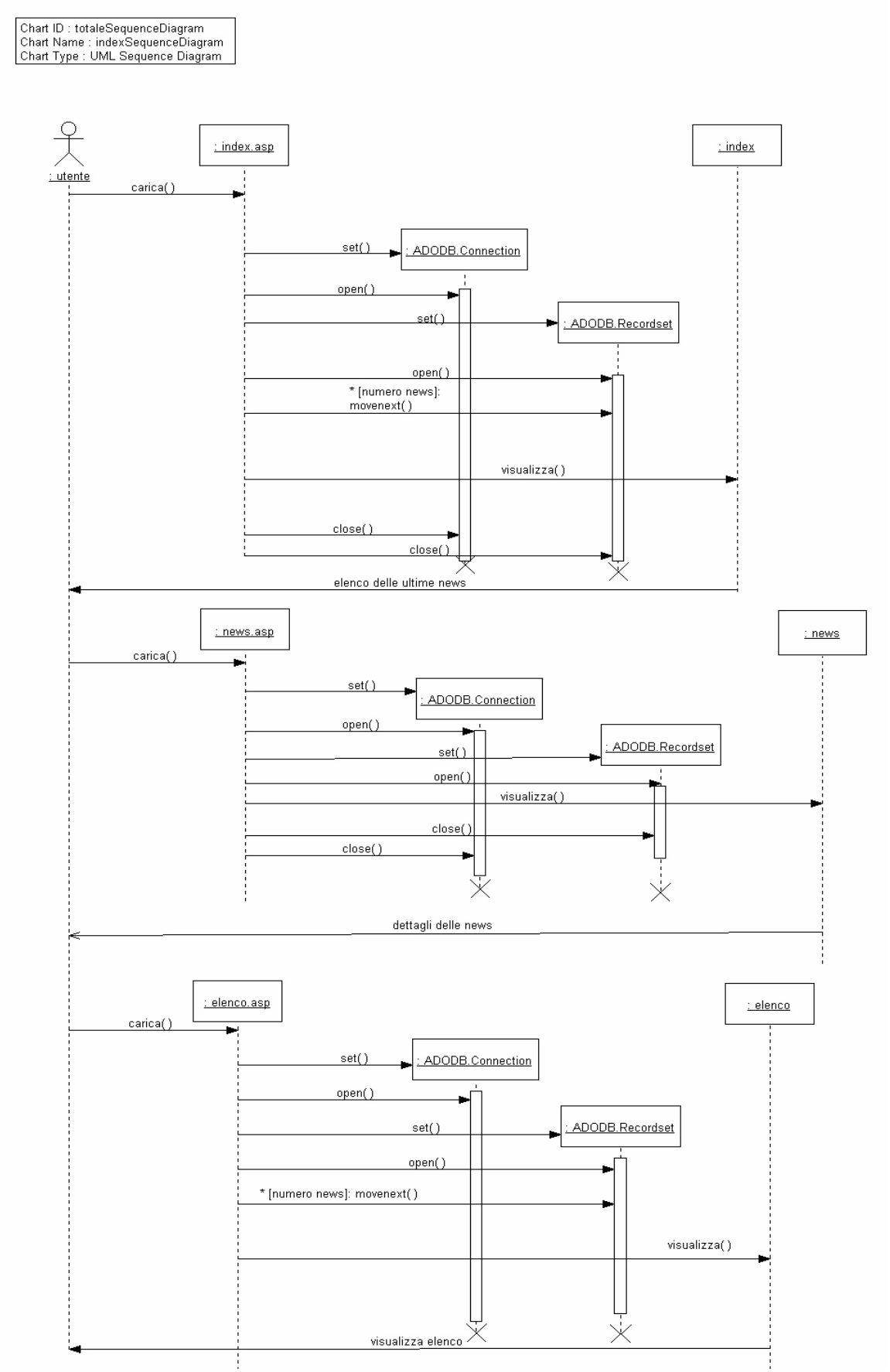


Figura 11 - Sequence diagram risultante dall'analisi dinamica del gruppo comprendente i file index.asp, elenco.asp e news.asp

2.4 Use case del sistema

L'analisi eseguita di questo sistema ha portato prima ad ipotizzare una suddivisione in sottosistemi, poi a individuare il particolare compito di ognuno di essi.
Si sono ricavati infine quattro diversi comportamenti del sistema e due tipologie di utenti. A questo punto si possono raccogliere tutte queste informazioni astraendo da esse un use case diagram del sistema, e quindi, per estensione, i suoi requisiti funzionali.

Use case aggiunta news

Si tratta di un'operazione che può essere fatta da un qualsiasi utente del sistema, e consiste nell'immettere alcuni dati personali, nonché il testo della news. Il sistema si limita ad aggiungere tali dati e il testo della news al database, rendendola quindi immediatamente disponibile a tutti.

Use case cancellazione news

Quest'operazione può essere fatta unicamente dall'amministratore del newsgroup, identificato da alcuni suoi dati personali, compreso un codice. Nell'applicazione tali dati sono compresi direttamente come costanti nella pagina, per cui bisogna supporre che la pagina sia accessibile direttamente solo dall'amministratore stesso.

Use case visualizzazione news dall'elenco completo

Quest'operazione può essere fatta da qualsiasi utente e consiste nel visualizzare l'elenco completo delle news, suddiviso in pagine, e visualizzare, a scelta, una news, corredata del nome dell'autore e di alcuni suoi dati personali.

Use case visualizzazione news dall'elenco delle più recenti

Anche quest'operazione viene fatta da qualsiasi utente e consiste nel visualizzare soltanto l'elenco delle news più recenti rispetto all'istante di immissione. Da quest'elenco è possibile giungere alla visualizzazione del testo di una news analogamente al caso precedente. Si può inoltre raggiungere l'elenco completo.

Attori del sistema

Si sono riconosciute due tipologie di utenti: utente generico, che ha il permesso di leggere o aggiungere news e amministratore, che ha in più la facoltà di cancellare

delle news. Queste considerazioni portano a definire una semplice gerarchia negli attori del sistema, dove si distingue una categoria generale *utente* che si partiziona nelle due categorie disgiunte *utente esterno* e *amministratore*.

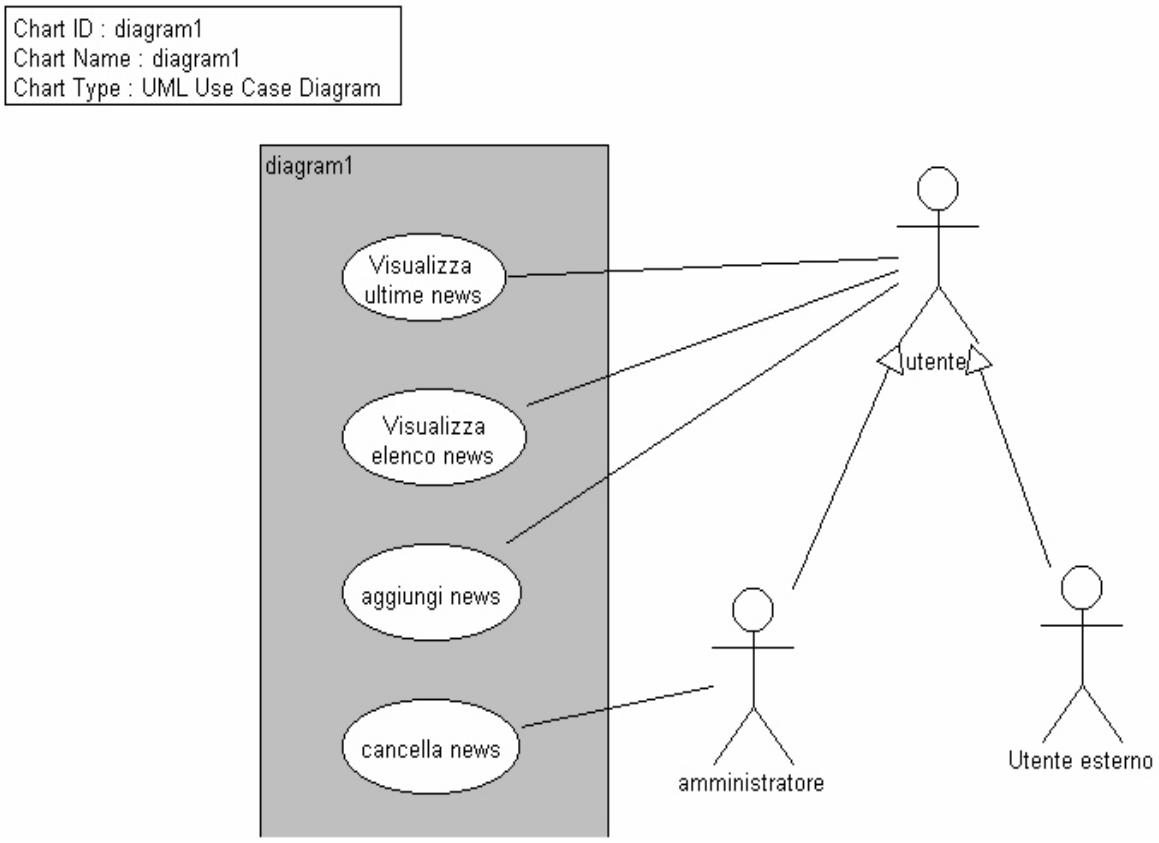


Figura 12 - Use case complessivo del sistema

CAPITOLO 6 - UNA METODOLOGIA DI REVERSE ENGINEERING PER APPLICAZIONI WEB

1. Una metodologia di reverse engineering per applicazioni web

In questo capitolo, sarà descritta una metodologia generale, per fornire alcune linee guida nel processo di reverse engineering di un'applicazione web. La metodologia è stata ricavata a partire dall'esempio analizzato nel precedente capitolo, in base quindi alla *lesson learned*.
Scopo di questa analisi è anche quello di giungere alla definizione di strumenti automatici per il reverse engineering.
Non si pretende ovviamente di poter dare una metodologia completa a partire dall'unico esempio studiato, peraltro estremamente semplice, ma di cercare di focalizzare l'attenzione su alcuni concetti basilari, in modo da poter ricavare dei modelli che siano ottenibili in una maniera quanto più automatica possibile.

Il processo di reverse engineering viene suddiviso in fasi. Tali fasi dovrebbero essere ripetute iterativamente, in modo da affinare ad ogni passo la comprensione del sistema, fino ad ottenere un risultato che sia ritenuto soddisfacente. L'ordine secondo cui saranno trattate le fasi é, quindi, quello preferenziale ma non è quello obbligatorio.
Le tre fasi principali del processo sono:

- Analisi statica;
- Analisi dinamica;
- Analisi funzionale.

L'input a tali fasi è rappresentato dal codice sorgente dell'applicazione web da analizzare e dall'applicazione stessa funzionante ed istallata su un web server.
Nella figura 1 è riportato un diagramma comprendente queste tre fasi con le rispettive interazioni. Nei prossimi paragrafi verranno descritte in dettaglio queste fasi presentando anche i modelli che risultano da ogni fase.

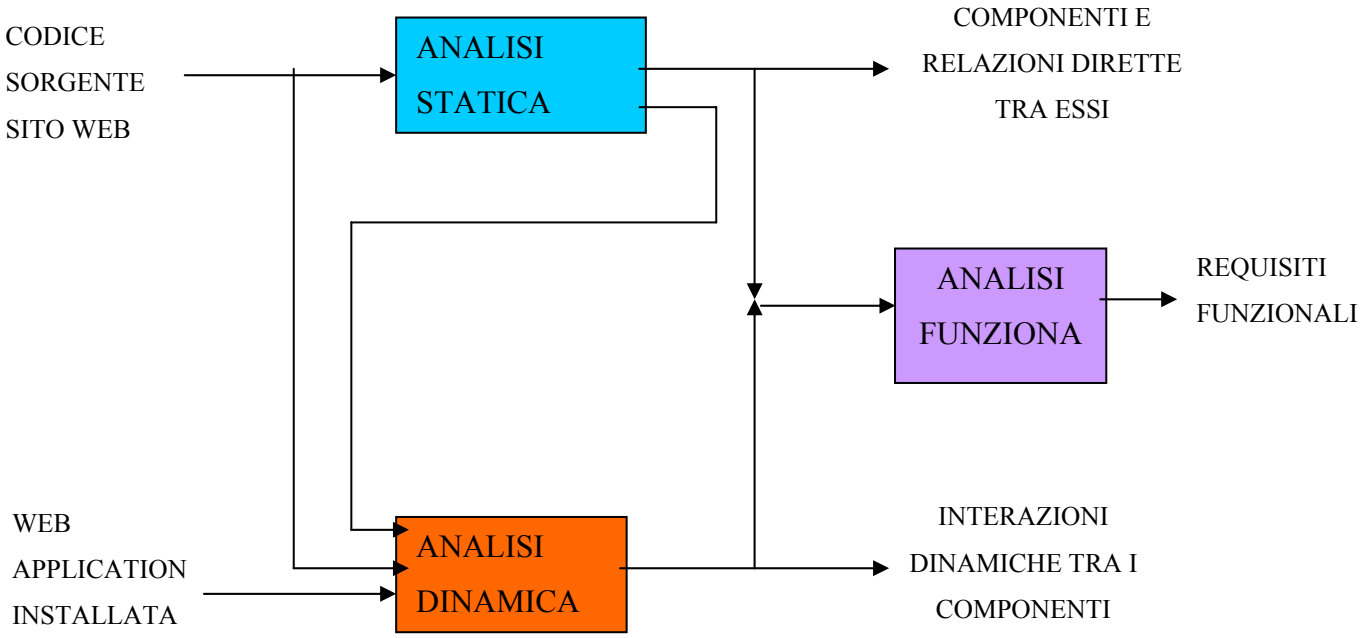


Figura 1 – Fasi della metodologia di reverse engineering di un'applicazione web

1.1 Fase 1: Analisi statica del sistema

Questa fase ha come input i file sorgenti costituenti l'applicazione da analizzare. Si effettua un'analisi sistematica di tutto il codice interpretabile dal browser o dal web server (pagine HTML, pagine server, moduli javascript, etc.). Scopo dell'analisi è di estrarre tutto ciò che può riguardare gli elementi componenti l'applicazione ed i collegamenti tra essi.

In questa fase si potrà rilevare solo una sottoparte delle interazioni esistenti, precisamente quelle statiche. Sono, infatti, possibili delle interazioni dinamiche, che vengono cioè esplicitate solo a tempo di esecuzione. In questa fase, sarà al massimo possibile prendere nota dell'esistenza di una tale interazione, senza però poterne conoscere la destinazione.

Le analisi degli altri file (ad es. le immagini, i database) si limiterà invece a registrarne l'esistenza in modo da poterli vedere come oggetti passivi delle connessioni partenti dalle pagine.

E' possibile individuare due "sottofasi" che compongono la fase di analisi statica:

- Organizzazione dei file in sottogruppi e scelta dell'ordine di analisi;
- Ricerca delle informazioni nel codice.

Nella prima sottofase viene fatta un'analisi preliminare del codice per individuare una suddivisione dell'applicazione web in sottogruppi. Si ipotizza che questi sottogruppi possano essere indipendenti tra loro e realizzare completamente una funzionalità. I criteri che si possono adottare per la suddivisione in sottogruppi sono puramente euristici (infatti la suddivisione proposta deve essere verificata nel prosieguo dell'analisi). La strategia che sembra più efficace consiste nel suddividere i file in base alle directory di appartenenza. In alternativa si può sfruttare la semantica dei nomi dei file.

Una volta fatta la suddivisione, bisogna scegliere l'ordine di analisi dei file, che può essere un fattore determinante nella riduzione del tempo di analisi, qualora essa non sia automatica. Una buona strategia consiste nell'iniziare l'analisi dalla pagina iniziale della navigazione (qualora sia nota) e procedere verso le pagine ad essa collegata, fino a chiudere una componente dell'applicazione(oppure l'applicazione intera qualora gli elementi in essa siano completamente raggiungibili).

La seconda sottofase è il nucleo dell'analisi statica. Analizzando il codice sorgente devono essere considerate quelle informazioni che indicano la presenza di un

collegamento tra due file contenenti pagine client e/o server e la presenza di elementi che consentano all'utente di influenzare l'esecuzione dell'applicazione (ad esempio una form HTML). Tali informazioni sono, in genere, molteplici e dipendono dalla tecnologia con cui l'applicazione web è implementata.

Le informazioni estratte saranno poi rappresentate in modo analitico, compilando tabelle, e in modo grafico, utilizzando diagrammi. Considerando l'esempio sviluppato nel capitolo precedente si possono indicare i seguenti modelli risultanti dall'analisi statica:

- Elenco dei componenti: è una lista comprendente tutti i file dell'applicazione web analizzati. E' utile per separare i componenti interni all'applicazione web da quelli esterni che non vengono analizzati.
- diagramma dei collegamenti: è un tipo di diagramma molto intuitivo e leggibile, ma ha il difetto di non riuscire a rappresentare una grossa fetta di collegamenti possibili, come quelli con immagini, database,etc. Come descritto nel capitolo 4, esso contiene come elementi di base le pagine client e/o server presenti nell'applicazione web e i collegamenti statici tra esse.
- class diagram UML: è un diagramma che contiene più informazioni rispetto al diagramma dei collegamenti. Un difetto però consiste nell'essere legato all'ambiente Object Oriented, verso cui UML è diretto. Come si è già visto nei capitoli precedenti, il modello UML non è adatto a poter rappresentare in maniera perfetta ed esaustiva l'ambiente delle applicazioni web. E' quindi necessario introdurre alcuni metodi fittizi allo scopo di visualizzare tutti i possibili collegamenti tra le classi. Inoltre si dovrà abusare un pò di quegli strumenti di estensione del linguaggio che sono gli stereotipi.

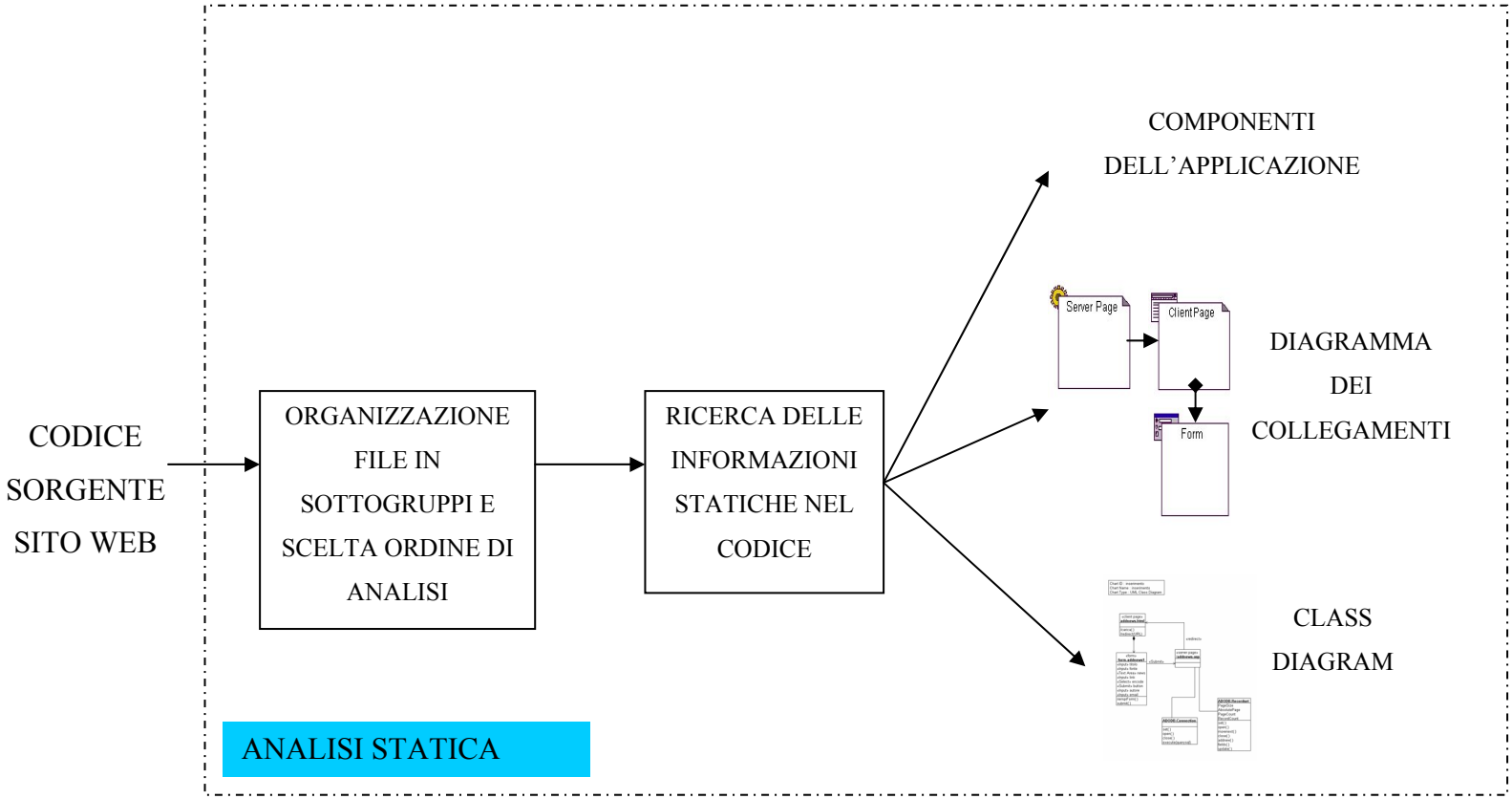


Figura 2 – Fase di analisi statica nella metodologia di reverse engineering di un'applicazione web

1.2 Fase 2: Analisi dinamica del sistema

In questa fase del processo si vuole ricavare informazioni sulle interazioni dinamiche tra i componenti del sistema, avendo come input i file sorgenti e le informazioni ottenute dall'analisi statica, nonchè osservando il sistema in esecuzione ed associando i vari eventi ad elementi del codice.

Mentre nella fase di analisi statica i file sorgenti vengono analizzati isolatamente, in questa fase si seguono i collegamenti tra le pagine client e/o le pagine server, definendo dei possibili scenari operativi. I collegamenti da seguire si possono ricavare dai modelli ottenuti dall'analisi statica del codice sorgente. Oltre alle pagine server e/o le pagine client in tali scenari saranno coinvolti anche altri elementi dell'applicazione web in esame come le form o gli oggetti interfacciati dalle pagine server. Nelle pagine server vengono considerate anche quelle parti di codice che risultano in un'elaborazione sequenziale.

In questa fase é sicuramente molto importante l'apporto della conoscenza ed esperienza umana, ragion per cui é difficile pensare a strumenti completamente automatici.

Risultato di tale fase saranno i sequence diagram di UML. Ad ogni sequence diagram corrisponde uno scenario operativo individuato dall'analisi del codice sorgente. In tale scenario si possono individuare più percorsi che sono associati ad eventi guidati dall'utente. L'inconveniente di tale rappresentazione é lo stesso dei class diagram, e dipende dal fatto che il mondo delle applicazioni web non é completamente rappresentabile nell'ambito delle metodologie Object Oriented. Per questo motivo é bene che ogni diagramma sia corredato da una descrizione che ne chiarisca l'esatta semantica.

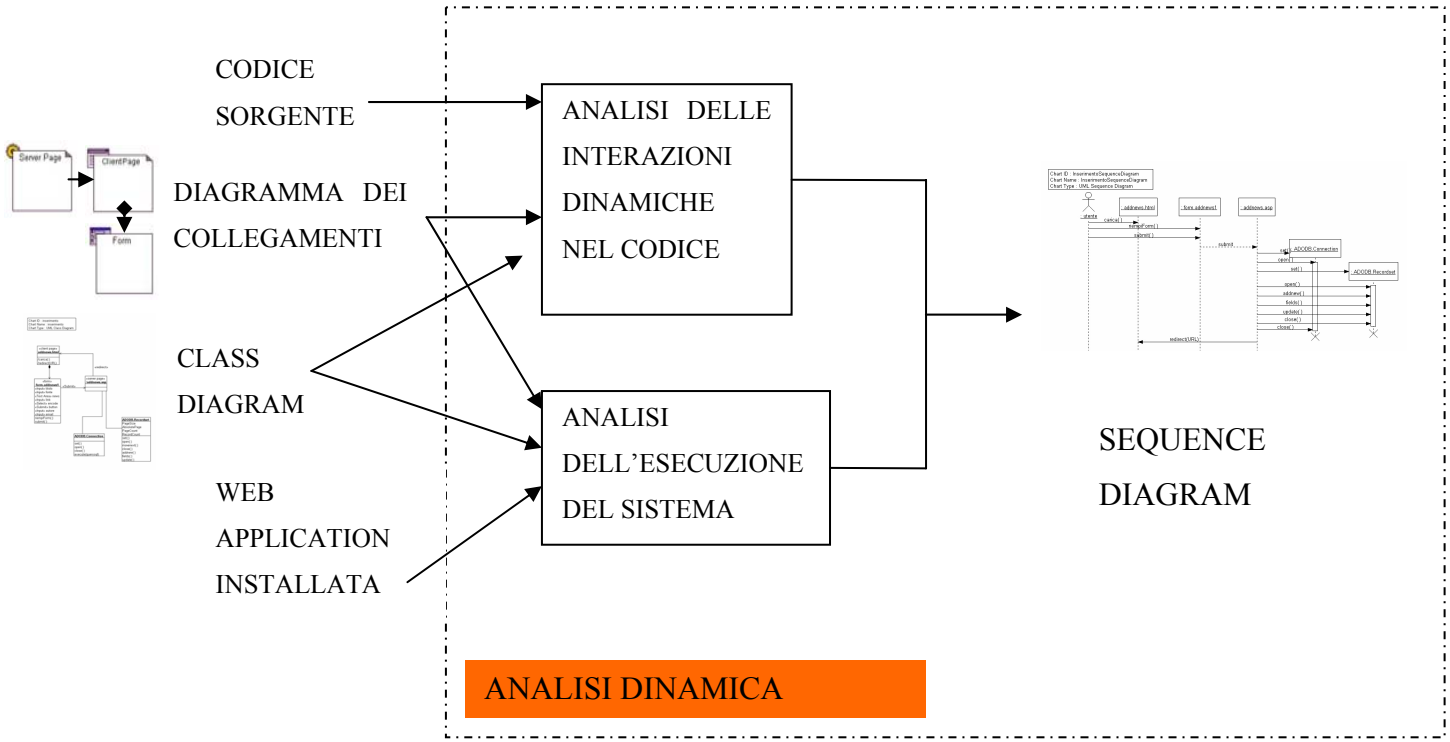


Figura 3 – Fase di analisi dinamica nella metodologia di reverse engineering di un'applicazione web

1.3 Fase 3: Analisi funzionale

In questa fase bisogna trarre delle conclusioni a partire dalle informazioni ricavate e dai diagrammi redatti nelle precedenti fasi. Si identificano le varie funzionalità del sistema, astraendole dalle informazioni ricavate in precedenza, e quali sono le componenti che le realizzano.

Uno dei principali obiettivi di questa fase è la suddivisione dell'applicazione web in macro-componenti (costituiti da opportune aggregazioni dei componenti precedentemente individuati), ognuna dei quali realizzi uno specifico caso d'uso.

Come input a questa fase si possono considerare i modelli ottenuti dall'analisi statica e da quella dinamica.

Partendo dai diagrammi ottenuti dalle analisi statica e dinamica, si può procedere all'interpretazione degli stessi, in una fase di "comprensione" della struttura dell'applicazione web. Scopo di tale fase è di poter ricavare delle componenti connesse del diagramma dei collegamenti, aiutandosi anche con gli altri diagrammi, le cui pagine server e/o client incluse implementano un'ipotetica funzionalità del sistema. Tali componenti connesse si possono associare ad una pagina origine, che si può supporre come punto di partenza della funzionalità.

Uno strumento automatico che può essere d'aiuto per valicare le ipotesi scaturite dalla comprensione, è quello che cerca di ricavare tutti gli oggetti che siano "raggiungibili" a partire da una pagina origine(ovvero tutti gli oggetti in essa inclusi, tutte le pagine ad essa collegate e tutti gli oggetti inclusi in queste altre pagine). Tali oggetti si possono riassumere in un "grafo di raggiungibilità" di una pagina.

Ad ogni funzionalità individuata con questa analisi, si associa un use case del sistema individuando anche gli attori partecipanti. L'individuazione degli attori non è semplice, ma può essere supportata da un'osservazione dell'esecuzione delle funzionalità e dei differenti profili di utente a cui sono rivolte.

Il modello riassuntivo che è possibile ricavare da tale analisi consiste negli use case diagram di UML. Tali diagrammi, estremamente semplificativi, devono essere corredati da descrizioni dettagliate che ne indicano anche il processo seguito per la loro redazione.

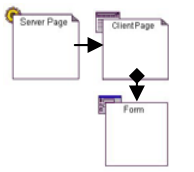


DIAGRAMMA DEI COLLEGAMENTI

CLASS DIAGRAM

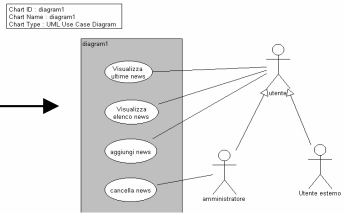
SEQUENCE DIAGRAM

COMPRENSIONE ED INDIVIDUAZIONE DI COMPONENTI CONNESSE

GRAFI DI RAGGIUNGIBILITA’

INDIVIDUAZIONE ATTORI

INDIVIDUAZIONE CASI D’USO



USE CASE DIAGRAM

ANALISI FUNZIONALE

CAPITOLO 7 - UN TOOL DI SUPPORTO AL REVERSE ENGINEERING

1.Introduzione

Come già accennato nel precedente capitolo, il processo di reverse engineering si avvale spesso del supporto di strumenti software per accompagnare le fasi individuate (astrazione ed estrazione).

Poiché l'obiettivo della fase di estrazione è quello di ricavare informazioni dai file sorgenti del sistema software da analizzare, gli strumenti software a supporto sono detti *estrattori* o *analizzatori*. L'analisi può restringere l'attenzione semplicemente alle informazioni statiche contenute nel codice sorgente (analisi statica), oppure alle informazioni legate all'esecuzione del sistema software (analisi dinamica). Queste ultime informazioni sono più difficilmente ricavabili da strumenti automatici che dovrebbero andare ad intercettare ciò che succede nell'esecuzione del sistema software in esame.

Nel caso delle applicazioni web, la fase di estrazione diventa complicata per la caratteristica eterogeneità di tali applicazioni. Infatti un'applicazione web comprende componenti che sono scritti in linguaggi ipertestuali o di scripting diversi. L'estrattore quindi deve poter riconoscere dal codice sorgente informazioni che siano associate ai numerosi linguaggi ipertestuali o di scripting presenti nell'applicazione.

Nella fase di astrazione gli strumenti software a supporto sono detti *astrattori*. Tali strumenti partono dalle informazioni ricavate dagli *estrattori* e le elaborano per ottenere astrazioni del sistema software in accordo con i modelli associati al processo di reverse engineering.

La fase di astrazione per le applicazioni web ha come obiettivo una serie di modelli che sono stati introdotti nei precedenti capitoli. Si può partire dai diagrammi dei collegamenti e dai grafi di raggiungibilità fino ad arrivare agli use case diagram che aiutano anche ad individuare le funzionalità dell'applicazione analizzata.

L'obiettivo di questo capitolo è di definire i requisiti di un sistema software a supporto del reverse engineering di applicazioni web. Si parte da un'analisi delle applicazioni di forward e reverse engineering descritte in letteratura. Tale analisi si può considerare come uno dei punti di partenza per la definizione dei requisiti del tool da sviluppare, visto che si vogliono anche individuare delle funzionalità che siano "originali". Il resto del capitolo è dedicato all'individuazione delle funzionalità del tool e alla descrizione dell'architettura.

2.Cenni allo stato attuale dell'arte

In questo paragrafo si vuole dare qualche informazione sui tool di supporto alla realizzazione e all'analisi delle applicazioni web.

Molte case produttrici di software hanno individuato nel web un campo in espansione e quindi si sono dedicate alla realizzazione di strumenti che possano consentire anche agli utenti meno esperti di costruire una applicazione web o, più semplicemente, un sito web. I prodotti in commercio sono molto differenti tra di loro e ciò è in accordo con la eterogeneità delle applicazioni web e dei componenti che le costituiscono.

I tool di supporto alla realizzazione o all'analisi di un'applicazione o un sito web si possono raggruppare in differenti categorie:

- Editor visuali per la gestione di siti web
- Tool dedicati alle presentazioni multimediali
- Tool che consentono di aggiungere interazione con un database
- Tool dedicati al reverse engineering

Nei paragrafi seguenti viene fornita una descrizione di ogni categoria individuata.

2.1 Editor visuali per la gestione di siti web

Tali tool sono un'evoluzione dei più semplici editor HTML. Non supportano lo sviluppo di grandi applicazioni web, ma soprattutto sono orientati alla gestione del sito web in cui sono incluse pagine web passive. Lo scopo di questi tool è quello di non costringere l'utente a dover scrivere codice in linguaggio HTML, ma di generare automaticamente codice a partire da oggetti visuali, secondo il diffuso paradigma WYSIWYG (What You See Is What You Get). Vengono anche fornite alcune funzionalità per la manutenzione del sito web. Tali funzionalità a volte si limitano semplicemente a presentare i collegamenti statici tra le pagine del sito web per presentare anche una gerarchia tra esse. Tra i tool più diffusi di questa categoria si possono citare NetObject Fusion, Microsoft FrontPage, Macromedia Dreamweaver.

2.2 Tool dedicati alle presentazioni multimediali

In questa categoria si possono raggruppare quei tool che sono nati con l'obiettivo di realizzare presentazioni multimediali off-line e che successivamente si sono estesi

anche alla piattaforma web. Sono quindi orientati alla realizzazione di pagine web statiche, ma con uno sguardo particolare alla multimedialità. Forniscono infatti facilitazioni per poter inserire in tali pagine oggetti multimediali come audio, video e per integrarli con il linguaggio HTML. Tra essi si possono citare Asymmetrix Toolbook, Macromedia director, Microsoft Publisher e Microsoft Power Point.

2.3 Tool che consentono di aggiungere interazione con un database o con altri oggetti

Fanno parte di questa categoria ambienti che permettono di aggiungere alle applicazioni web estensioni rispetto al linguaggio HTML di base. In genere sono editor del linguaggio di scripting che realizza la dinamicità delle pagine web secondo lo schema architetturale "thick web client". Un difetto di tali tool è che non consentono di definire una struttura del sito web e a volte si limitano alla realizzazione della singola pagina web. Si possono includere in tale categoria gli editor Javascript, Asp e di qualsiasi altro linguaggio di scripting incluso nelle applicazioni web.

2.4 Tool di reverse engineering

In letteratura è stato trovato un solo esempio di un tool dedicato al reverse engineering di applicazioni web: ReWeb [15]. Tale tool è effettivamente orientato ad un'analisi del cambiamento delle applicazioni web.

ReWeb lavora su tutte le pagine incluse in un dominio, il cui indirizzo di partenza è specificato come input. Modella le connessioni tra le pagine del sito web utilizzando un formalismo basato su grafi. L'obiettivo principale del tool, oltre la visualizzazione della struttura del sito, è il confronto tra differenti versioni di esso, al fine di studiarne l'evoluzione temporale.

La versione attualmente disponibile si occupa unicamente delle interazioni tra pagine statiche (scritte completamente in linguaggio HTML).

2.5 Conclusioni

Gli strumenti software che sono stati introdotti non accompagnano l'utente in un processo di sviluppo dell'applicazione web ben definito. Ciò è anche conseguenza del bacino di utenza cui questi software si rivolgono, che è costiuito anche da utenti poco esperti. Inoltre si riscontra in letteratura anche una mancanza di modelli formali completi e ben consolidati per la descrizione sia dell'applicazione web, sia del suo processo di sviluppo.

In particolare il problema della modellizzazione del comportamento delle pagine dinamiche in un'applicazione web non viene affrontato, se non con le metodologie tipiche delle applicazioni object oriented, il cui adattamento crea delle carenze semantiche.

Per tale motivo è difficile trovare uno strumento che abbia l'obiettivo di ricavare modelli dalle applicazioni web esistenti.

3.Analisi dei requisiti del tool

3.1 Introduzione

Nei paragrafi successivi è riportata l'analisi dei requisiti del tool di supporto al reverse engineering delle applicazioni web. L'analisi è organizzata seguendo lo standard IEEE 830 del 1984.

3.2 Campo di applicazione

Il sistema da sviluppare è rivolto alle attività di Reverse Engineering di un'applicazione web.

3.3 Scopo del sistema

Lo scopo del sistema è quello di eseguire in maniera automatica sia la fase di estrazione delle informazioni associate all'applicazione web che quella di astrazione per tracciare i modelli individuati. Tali modelli danno una rappresentazione delle entità che compongono l'applicazione web e delle interazioni tra esse. Con il termine entità ci si riferisce a tutti gli elementi che costituiscono l'applicazione web a partire dalle singole pagine fino agli oggetti con cui tali pagine interagiscono.

In questo modo si vuole fornire uno strumento di reverse engineering quanto più possibile automatico che possa aiutare a capire come è strutturata l'applicazione in esame a supporto anche di un'attività di manutenzione della stessa. Un altro obiettivo é quello di andare ad analizzare la qualità dell'applicazione in esame e la navigabilità all'interno delle pagine che la compongono. Il sistema può anche essere di supporto ad un'attività di testing dell'applicazione web, in quanto è cruciale in questa attività capire come l'applicazione è effettivamente strutturata.

3.4 Descrizione generale

In questa sezione si iniziano ad analizzare le funzionalità del sistema in generale, per poi dividere l'analisi ai vari componenti del sistema che si andranno ad interfacciare.

3.4.1 Funzionalità esterne del sistema

Il sistema da realizzare, per ogni pagina che compone l'applicazione web da analizzare, deve riportare tutte le entità in essa contenute e tutti i riferimenti ad altre entità secondo i modelli e le astrazioni presentate nel capitolo precedente. Tali informazioni devono essere memorizzate in un database di supporto che viene messo a disposizione dell'utente.

Il database sarà opportunamente interrogato per ottenere rappresentazioni testuali e/o grafiche della applicazione di interesse.

3.4.2 Caratteristiche dell'utente

L'utente del sistema deve avere a disposizione una applicazione web da analizzare. Può trattarsi, quindi, di un team che deve realizzare un'attività di manutenzione di una applicazione web, un team per il testing, un team per l'analisi di qualità dell'applicazione o qualsiasi altra tipologia di utente interessata ad avere una rappresentazione dell'applicazione da analizzare ad un più alto livello rispetto a quello della sua implementazione.

4 Architettura del sistema

Lo scopo di questo paragrafo è quello di individuare i componenti del sistema che possano avere uno sviluppo separato e per questi andare a individuare le funzionalità generali.

Dalla definizione dei requisiti si può individuare come primo componente l'*estrattore* che provvede a ricavare le informazioni contenute nei file sorgenti dell'applicazione web in analisi, che rappresentano quindi l'input del sistema completo.

Le informazioni ricavate possono essere presentate in una *forma di rappresentazione intermedia* in modo che l'utente possa analizzarle e un software possa inserirle in un opportuno *database*, per costruire i modelli di output del sistema completo.

Si individua, quindi, un *astrattore di I livello* che provvede ad inserire le informazioni nel database. Il database rappresenta infatti già una prima astrazione dell'applicazione web , anche perché la presenza di opportune query consentirà la navigazione tra le tabelle del database e la possibilità di ricavare le prime informazioni sui possibili collegamenti esistenti tra i componenti dell'applicazione web in esame.

Per presentare i modelli conclusivi dell'applicazione web c'è bisogno di un software di visualizzazione che consenta anche l'interazione con l'utente. Tale software si può denominare *astrattore di II livello* visto che realizza un'astrazione di livello maggiore rispetto al database.

Nella figura seguente è presentata l'architettura completa del sistema con i componenti individuati.

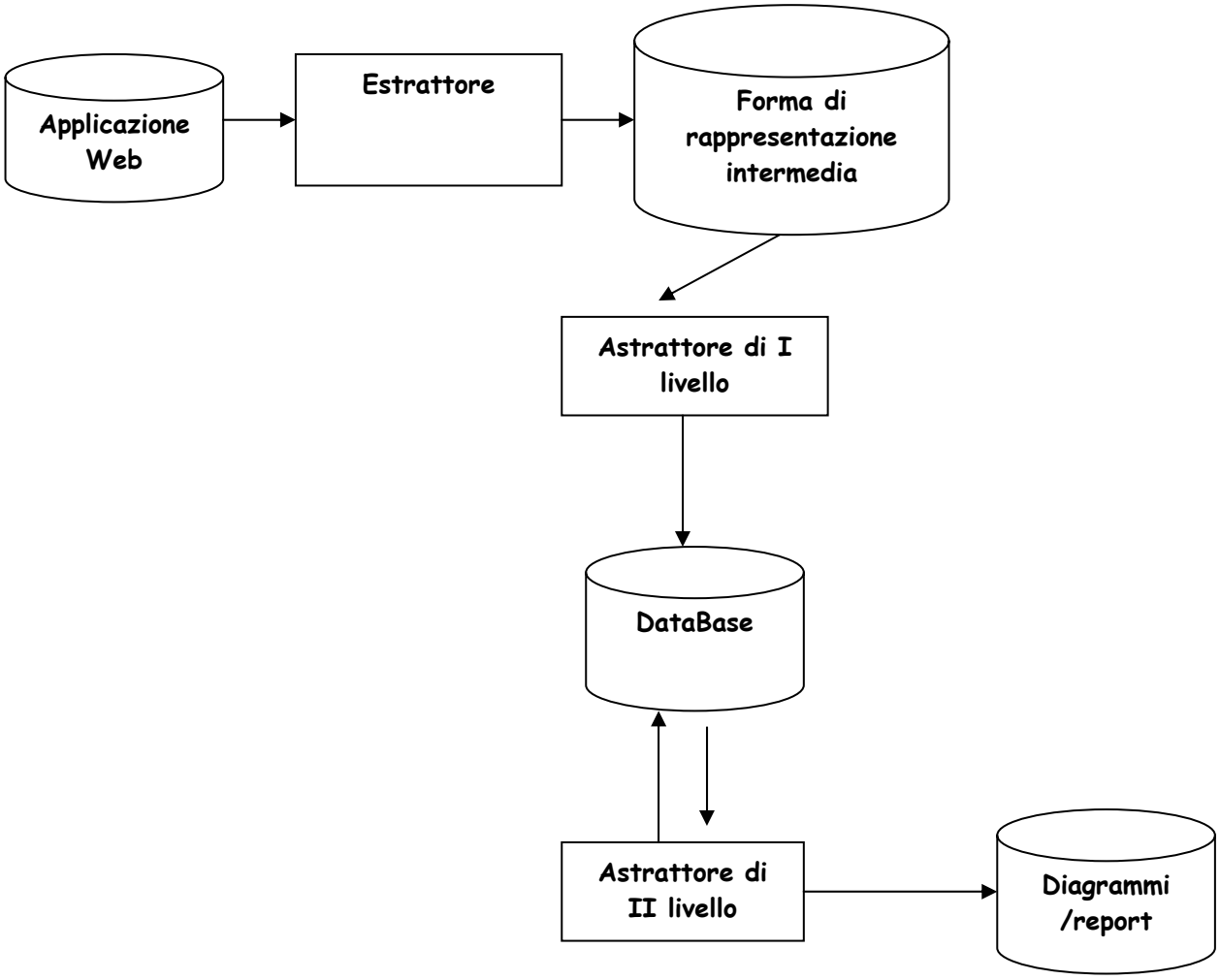


Figura 1 Rappresentazione dell'architettura del tool

Nei paragrafi successivi viene data una descrizione dettagliata dei componenti del sistema e per ognuno di essi si tenta di darne un'iniziale specifica dei requisiti.

4.1 Applicazione Web

Rappresenta l'input al sistema. E' composta da tutti i sorgenti dell'applicazione web da analizzare. I sorgenti devono mantenere la stessa posizione relativa rispetto alla directory principale dell'applicazione in esame, altrimenti le informazioni ricavate possono essere errate.

4.2 Estrattore

Tale componente ha come scopo quello di riconoscere i tag del linguaggio HTML e dei linguaggi di scripting supportati. In sede di realizzazione dell'estrattore si evidenzierà il relativo campo di applicazione.

Come già introdotto in precedenza, l'output dell'estrattore deve essere una forma di rappresentazione intermedia. Si sceglie come forma intermedia un file "taggato" che racchiude, per ogni file sorgente analizzato, le informazioni estratte. Tale scelta viene qui anticipata, perché è fondamentale per definire i requisiti dell'estrattore conoscere la sintassi di tale file "taggato"

4.3 Database

Tale componente si può definire come il punto centrale del progetto. Dalla realizzazione del database si ricavano i requisiti dei componenti del sistema che hanno come obiettivo l'astrazione dell'applicazione web in analisi. Si può dire che il database è il punto di separazione tra la fase di estrazione e quella di astrazione.

4.4 Astrattore di I livello

Tale componente ha come input i file "taggati" ottenuti dai parser e ha lo scopo di interpretarne la sintassi e di memorizzare le informazioni ottenute nel Database.

4.5 Astrattore di II livello

Tale componente ha lo scopo di visualizzare le informazioni contenute nel database mediante i modelli presentati nei precedenti capitoli. Il software dovrà consentire l'interazione con l'utente e la possibilità di visualizzare le tabelle del database o i grafi che sono ottenuti da tali tabelle.

5 Vincoli sul campo di applicazione

A causa della forte eterogeneità dei componenti presenti in un'applicazione web e dei differenti linguaggi di scripting che si possono usare si vuole restringere il campo di applicazione del tool completo.

La restrizione riguarda in pratica le informazioni che vengono ricavate dal software estrattore.

Si vogliono prendere in considerazione applicazioni web che abbiano funzionalità dal lato client e dal lato server secondo l'architettura "thick web client" con le seguenti limitazioni:

- nelle pagine client vengono individuati (oltre ai tag HTML) solo blocchi di istruzioni scritte in linguaggio Javascript
- come pagine server si considerano solo quelle della tecnologia ASP in cui vengono individuati blocchi di istruzioni scritte in linguaggio VBScript.

Questo è il vincolo più generale del tool completo, per quanto riguarda i dettagli sui tag HTML riconosciuti e sulle istruzioni Javascript e VBScript si rimanda all'analisi dei requisiti del tool estrattore e del database.

6 Definizione del progetto

La suddivisione dell'architettura in componenti distinti, facilita un parallelismo nella realizzazione del tool completo.

Nella figura 2 è raffigurato il grafo delle attività necessarie alla realizzazione del tool estrattore.

Il primo passo consiste nell'analisi delle informazioni nel dominio applicativo indicato nel paragrafo 5 di questo capitolo. Dopo quest'analisi preliminare è possibile una definizione dei requisiti del tool estrattore. Una volta fissati, precisamente, i requisiti dell'estrattore si può scendere più in dettaglio definendo, in parallelo, la struttura del database e la sintassi e la semantica del file "taggato" che costituisce l'output dell'analisi di un file sorgente.

A questo punto sono disponibili tutti gli elementi per poter definire precisamente le specifiche dell'estrattore, e per poterlo implementare.

Il passo finale consiste nella verifica e validazione del tool, che si basa sull'analisi, in reverse engineering, di due casi di studio, seguendo la metodologia descritta nel capitolo 6, assistita dal tool automatico estrattore.

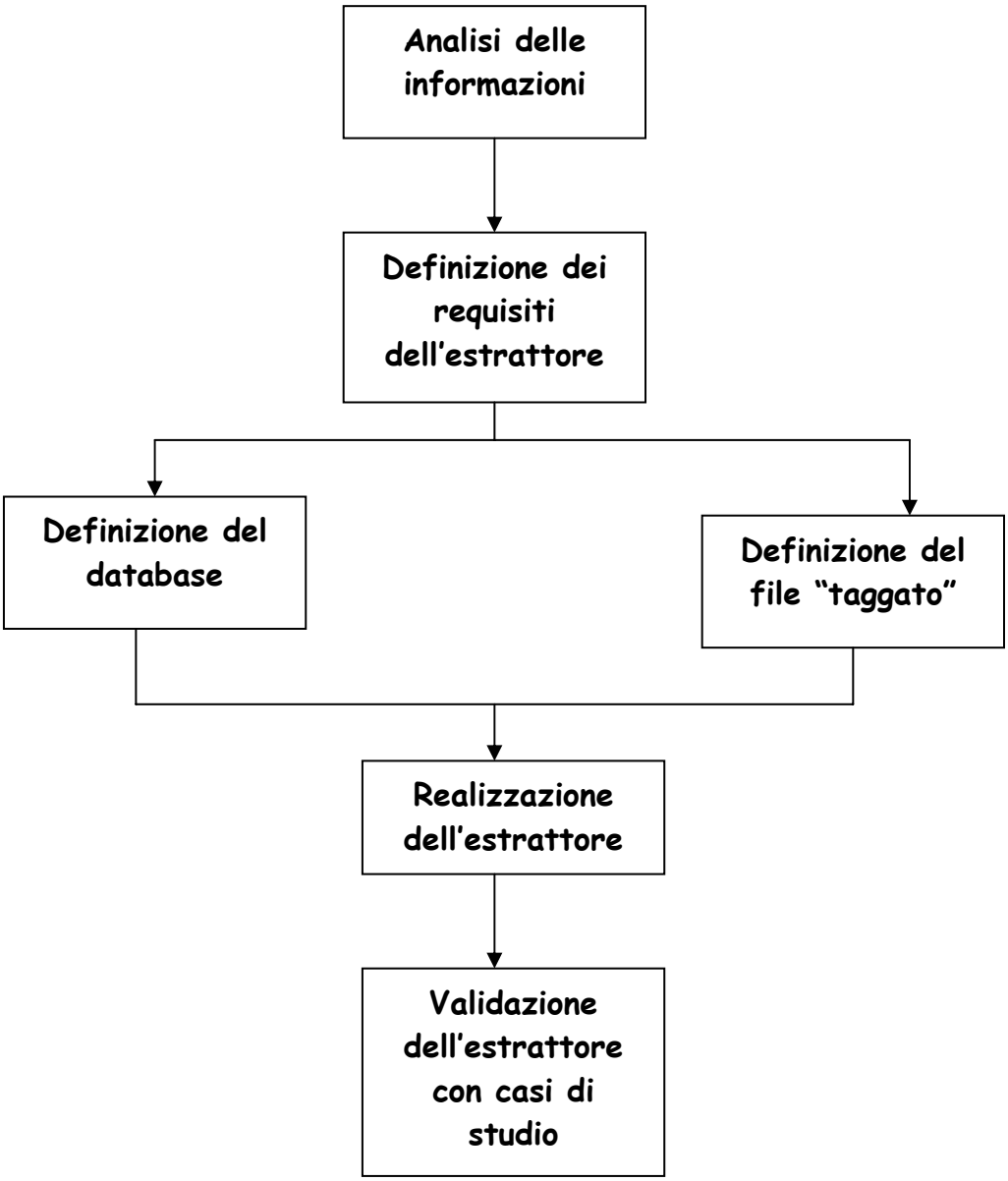


Figura 2 – Grafo delle attività

CAPITOLO 8 - IL DATABASE UTILIZZATO

1 Analisi dei requisiti per il database

1.1 Premessa

In questo capitolo sono riportate tutte le informazioni in base alle quali si è ottenuto il modello concettuale del database. Nel modello concettuale, realizzato con un class diagram UML, per una sua migliore leggibilità si sono omessi la maggior parte dei particolari riguardanti gli attributi di ogni singola entità o relazione, che pure si ritrovano nel modello relazionale sviluppato.

La base di dati che si vuole realizzare deve riportare tutte le informazioni relative a elementi, collegamenti e interazioni tra gli oggetti presenti in un'applicazione web sviluppata con le tecnologie ASP, HTML e Javascript.

1.2 Descrizione del campo di applicazione del database

Un'applicazione web consta di una serie di *pagine web* e di altri oggetti che possono essere inclusi in essi o interagire.
Si distinguono due famiglie di pagine web: le *pagine server*, le quali hanno estensione .asp e le *pagine client* che hanno invece estensione .htm (o .html). Di tali pagine si vuole conoscere il nome del file sorgente contenenti il codice.
Le pagine server possono comprendere al loro interno alcuni oggetti peculiari, ovvero *blocchi* di codice in linguaggio *VBScript*. Di tali blocchi dovrà essere specificata la posizione all'interno del file sorgente e il legame con il file sorgente in cui sono contenuti. I blocchi VBScript possono anche avere al loro interno un'istruzione di redirect che sposta il controllo dell'applicazione ad un'altra pagina web. Di tale istruzione si vuole conoscere la posizione all'interno del file sorgente.
Le pagine server possono interagire con degli oggetti esterni tramite delle interfacce. Tra le interfacce possibili si distinguono *interfacce ADODB Connection* e *Recordset* verso un database ODBC e *interfacce CDONTS* verso messaggi di posta elettronica.
Le pagine server possono costruire (*builds*) delle pagine client che vengono viste dall'utente ma non vengono memorizzate in alcun file. Anche tali pagine possono essere considerate come pagine web.

Le pagine client si dividono in due tipologie: *pagine client costruite*, ovvero quelle create dall'esecuzione delle pagine server e *pagine HTML*. Le pagine client sono le uniche a poter includere dei *blocchi di linguaggio Javascript* introdotti da un particolare tag HTML. Di tali blocchi, analogamente ai blocchi VBScript, si vuole conoscere la posizione all'interno del file sorgente. Tra i comandi possibili in questo linguaggio vi è il redirect che può spostare la navigazione verso un'altra pagina web qualsiasi. Anche di tale comando si vuole conoscere la posizione all'interno del file sorgente.

Esistono numerosi altri oggetti che possono essere presenti in ogni tipo di pagina client, che sono introdotti da un particolare tag HTML di cui si vuole conoscere la posizione nel file sorgente:

- *Ancore o riferimenti ipertestuali (Href),* le quali realizzano la struttura di link ipertestuale verso un'altra pagina web di qualsivoglia natura oppure si possono riferire a :
  - *Immagini visualizzabili*, come ad esempio files con estensione .gif, .jpeg, che vengono eseguiti automaticamente dal browser all'interno della pagina web;
  - *Altri oggetti,* di natura differente dalle immagini, che sono collegati mediante un link ipertestuale e vengono aperti solo se cliccati e non nella stessa finestra;

- *Form di input*, i quali rappresentano di punti di interazione con l'utente e sono dotati di un'operazione di submit che li collega ad un'altra pagina web oppure ad un'applicazione; essi sono dotati anche di parametri che devono essere specificati nella base di dati.

- *Frameset*, che sono particolari tag inclusi in una pagina web che indicano che tale pagina è suddivisa in più *frame*, ognuno dei quali è a sua volta una pagina web. Di tale frame si vuole specificare la pagina web sorgente e il frameset in cui è incluso. Si adotta la semplificazione che una pagina web può contenere al più un tag frameset in cui si considera incluso ogni tag frame.

- *Applet*, i quali servono ad includere in una qualsiasi pagina web un programma sviluppato in linguaggio Java e compilato in un file bytecode (con estensione .class). Tale oggetto è denominato *Java class*.

- *Inclusione modulo Javascript*, col quale è possibile richiamare un modulo scritto in linguaggio Javascript e contenuto in un apposito file (con estensione .js).

Si vuole lasciare poi ad un eventuale utilizzatore del database la possibilità di inserire delle annotazioni riferite agli oggetti o alle interazioni.
Inoltre si vuole lasciare la possibilità ad un eventuale programma che popoli automaticamente il database di inserire dei messaggi di *warning* in caso di problemi o di situazioni ambigue.
Infine, per ogni oggetto che sia contenuto in un file, si vuole poter tenere memoria del fatto che tale file appartenga al dominio analizzato oppure sia esterno ad esso.

2 Modello concettuale del database

E' stata fatta una panoramica sulle possibili entità che si hanno in una applicazione web. Tale panoramica è servita come punto di partenza per avviare un modello concettuale del database da realizzare. Tale modello è stato condotto utilizzando il formalismo dei class diagram del linguaggio UML. In particolare, ad ogni entità di interesse è stata associata una classe e ad ogni attributo dell'entità un attributo della classe. Ad ogni collegamento tra gli elementi individuati è stata associata una *relationship* del linguaggio UML e ne è stato specificato il tipo. In tali relazioni possiamo avere degli attributi specifici che sono stati assegnati ad una classe associativa.

Risultato è stato il class diagram di figura 1.

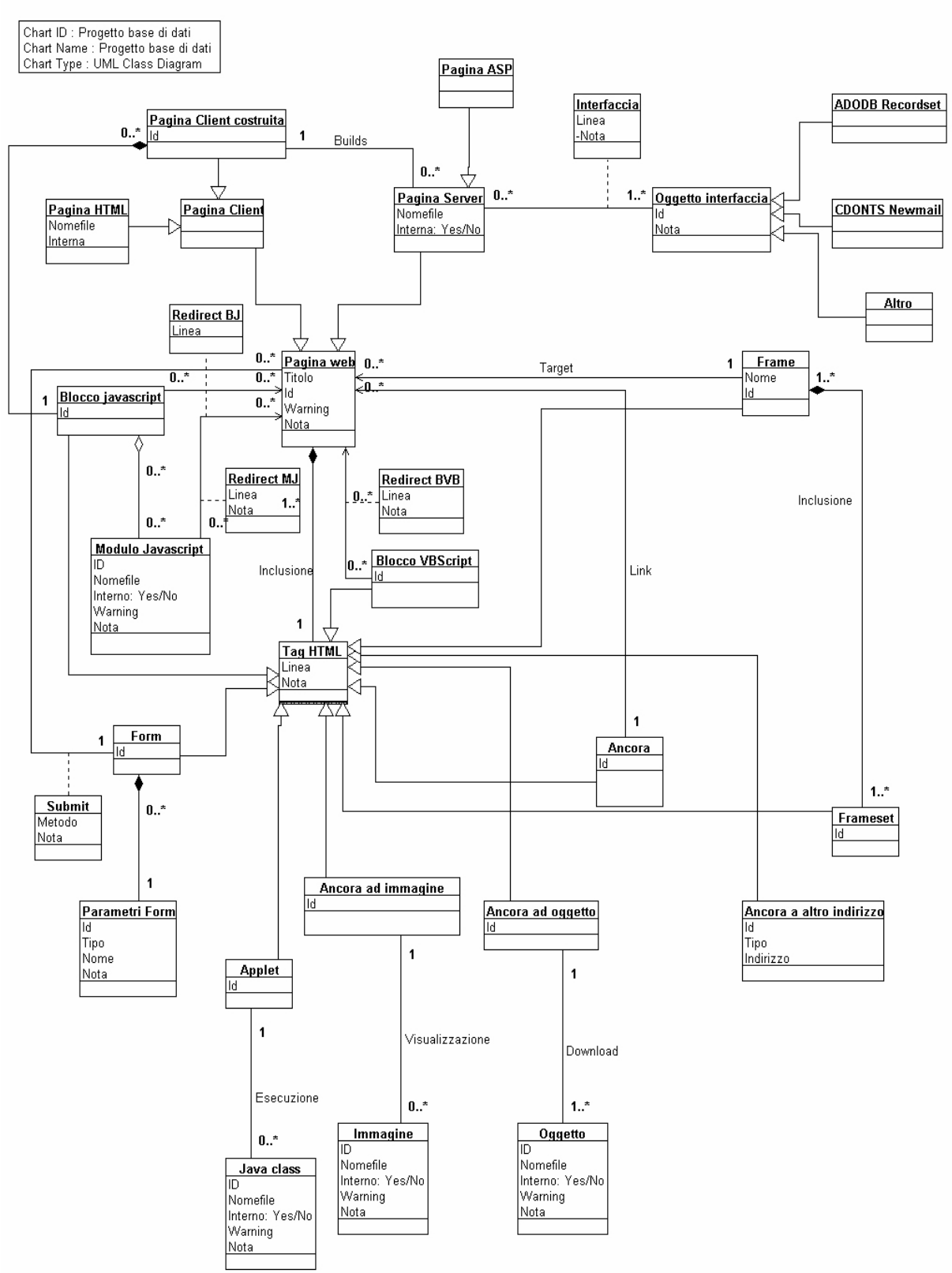


Figura 1 – Modello concettuale del database

3 Dizionario dei dati

In questo paragrafo si vogliono elencare tutti i nomi che compaiono nel modello del database, allegando una breve nota esplicativa ad ognuno di essi. In seguito si farà riferimento alle classi ed associazioni che corrispondono alle classi e alle relationship individuate nel modello UML. Per ogni classe vengono specificati gli attributi, mentre le associazioni a cui partecipa vengono descritte in seguito.
Inoltre, poiché lo scopo di questo modello è la sua successiva traduzione verso un modello relazionale di un database, viene anche dichiarato esplicitamente (facendo uso di stereotipi), quali attributi siano da considerarsi identificatori per le classi. Gli attributi identificatori sono indicati in grassetto.

3.1 Classi

Pagina web: nome dato ad ogni file scritto in un formato interpretabile da un browser o da un server. Le pagine web concentrano in esse tutto il contenuto interattivo di un'applicazione web.
Attributi:

- Titolo: ovvero il testo contenuto tra i tag <TITLE> e </TITLE>
- **Id**: numero seriale della pagina. Si tratta di un codice numerico che identifica univocamente una pagina
- Warning: indica che non si è riuscito ad associare alla pagina web un nome di file consistente
- Nota: spazio di commento per l'utente

Associazioni:

- Submit, con Form
- Redirect con Blocco Javascript
- Redirect con Modulo Javascript
- Redirect con Blocco VBScript
- Inclusione con Tag HTML (relazione di tipo composition)
- Link con Ancora

- Target con Frame

Specializzazioni:

- Pagina Client
- Pagina Server

La gerarchia di generalizzazione non è completa.

Pagina Client: si tratta di un'entità interpretabile dal browser presente sul computer client.

Specializzazioni:

- Pagina HTML
- Pagina Client costruita

Generalizzazioni:

- Pagina web

Pagina HTML: si tratta di una pagina client che esiste in maniera stabile su di un file con estensione .htm.

Attributi:

- **Nomefile**: col quale si intende il percorso completo cui è possibile reperire il file
- Interno: esprime la proprietà del file di essere contenuto nell'applicazione web oppure esterno ad esso.

Generalizzazioni:

- Pagina Client

Pagina Client costruita: si tratta dell'output risultante dall'esecuzione sul server di una pagina ASP, inviato ad un browser residente sul computer client.

Attributi:

- **Id**: numero seriale

Associazioni:

- Builds con Pagina Server

Generalizzazioni:

- Pagina Client

Pagina Server: si tratta di una pagina eseguibile da un web server.

Attributi:

- **Nomefile**: col quale si intende il percorso completo cui è possibile reperire il file
- Interno: esprime la proprietà del file di essere contenuto nell'applicazione web oppure esterno ad esso.

Associazioni:

- Builds con Pagina Client costruita
- Interfaccia con oggetto interfaccia

Generalizzazioni:

- Pagina web

Specializzazioni :

- Pagina ASP

Ovviamente tale gerarchia di generalizzazione non è completa.

Pagina ASP: si tratta di una pagina server realizzata con questa particolare tecnologia. La distinzione rispetto a Pagina Server è fatta solo per questo particolare tipo, coerentemente con il campo di applicazione dichiarato.

Generalizzazioni:

- Pagina Server

Oggetto interfaccia: si riferisce ad un qualsiasi oggetto col quale possa interagire una pagina ASP, il cui interfacciamento sia dichiarato con un'istruzione del tipo set. Nei casi in cui non sia possibile decidere staticamente il tipo dell'oggetto, quest'ultimo si intende come generico.

Attributi:

- **Id**: numero seriale
- Nota: spazio di commento per l'utente

Associazioni:

- Interfaccia con Pagina Server

Specializzazioni:

- ADODB Recordset
- ADODB Connection

- CDONTS.Newmail

Ovviamente questa gerarchia di generalizzazione non è completa.

ADODB Recordset: si riferisce al particolare oggetto che si interfaccia con la pagina web per la gestione dei record di un database

Generalizzazioni:

- Oggetto interfaccia

ADODB Connection: si riferisce al particolare oggetto che si interfaccia con la pagina web per la gestione della connessione ad un database

Generalizzazioni:

- Oggetto interfaccia

CDONTS Newmail: si riferisce al particolare oggetto che si interfaccia con la pagina web per la gestione dell'invio di una e-mail

Generalizzazioni:

- Oggetto interfaccia

Blocco Javascript: si tratta di un frammento di codice, incorporato in un tag HTML interpretabile dalla maggior parte dei moderni browser. Essendo fisicamente incorporato in un tag HTML, è imprescindibile dalla pagina nella quale è situato, per cui è stato introdotto un numero seriale come identificatore.

Attributi:

- **Id**: numero seriale

Associazioni:

- Redirect con una pagina web
- Inclusione con un modulo javascript

Generalizzazioni:

- Tag HTML

Modulo Javascript: si tratta di un frammento di codice javascript, il quale è però salvato in un file a sé stante, di modo che sia richiamabile da un blocco javascript in modo da garantire un riuso, sia pur elementare.

Attributi:

- **Id**: numero seriale
- Nomefile: col quale si intende il percorso completo cui è possibile reperire il file
- Warning: indica che non si è riuscito ad associare al modulo un nome di file consistente
- Nota: spazio di commento per l'utente

Associazioni:

- Redirect con una pagina web
- Inclusione con un blocco javascript

Tag HTML: quest'entità rappresenta un qualsiasi costrutto sintattico contenuto in una pagina in linguaggio HTML, che inizia con < *tag* >e termina con </ *tag*>.

Attributi:

- **Linea**: linea del file sorgente in cui è presente il tag
- Nota: spazio di commento per l'utente

Associazioni:

- Inclusione con una Pagina web (associazione di tipo composition)

Specializzazioni

- Blocco javascript
- Form
- Applet
- Ancora ad immagine
- Ancora ad oggetto
- Ancora ad altro indirizzo
- Ancora
- Frameset
- Blocco VBScript

Questa gerarchia di generalizzazione, nell'ambito del campo di applicazione previsto, è da considerarsi completa.

Form: si tratta di un tag che serve a realizzare un'interazione generica con l'utente. Possiamo avere dei tag di input/output (i più diffusi) oppure anche di solo output (in tal caso la relazione submit ha cardinalità zero),

Attributi:

- **Id**: numero seriale

Associazioni:

- Submit, verso una pagina web
- Inclusione, con Parametri Form (relazione di tipo composition)

Generalizzazioni:

- Tag HTML

Parametri Form: contiene i parametri contenuti in una singola form.

Attributi:

- **Id**: numero seriale
- Tipo: tipo del parametro.
- Nome: nome del parametro

Associazioni:

- Inclusione con una form (relazione di tipo composition)

Ancora: si tratta di uno dei casi del tag <A. Si è scelto di indicare semplicemente con ancora il caso classico di collegamento ipertestuale unilaterale tra due pagine web.

Attributi:

- **Id**: numero seriale

Associazioni:

- Link con una pagina web

Generalizzazioni:

- Tag HTML

Ancora ad immagine: si tratta di un caso particolare di ancora, nella quale l'oggetto coinvolto è un'immagine, in formato GIF o JPG. In tali casi, all'esecuzione della pagina web, il browser cercherà di caricare l'oggetto e di visualizzarlo nella pagina.

Attributi:

- **Id**: numero seriale

Associazioni:

- Visualizzazione con un'immagine

Generalizzazioni:

- Tag HTML

Immagine: una qualsiasi immagine in uno dei formati interpretabili dal browser.

Attributi:

- **Id**: numero seriale
- Nomefile col quale si intende il percorso completo cui è possibile reperire il file
- Warning: indica che non si è riuscito ad associare all'immagine un nome di file consistente
- Nota: spazio di commento per l'utente

Associazioni:

- Visualizzazione con ancora ad immagine

Ancora ad oggetto: si tratta di un caso particolare di ancora. La destinazione è un file che non sia un'immagine. Solo se l'utente cliccherà sull'ancora il browser provvederà a mandare una richiesta per il download del file specificato.

Attributi:

- **Id**: numero seriale

Associazioni:

- Download con un oggetto

Generalizzazioni:

- Tag HTML

Oggetto: un qualsiasi oggetto in uno dei formati non interpretabili dal browser.

Attributi:

- **Id**: numero seriale
- Nomefile: col quale si intende il percorso completo cui è possibile reperire il file

- Warning: indica che non si è potuto associare all'oggetto un nome di file valido
- Nota: spazio di commento per l'utente

Associazioni:

- Download, con un'ancora ad oggetto

Ancora ad altro indirizzo: in questa tipologia ricadono tutti i collegamenti ad indirizzi che non siano né del tipo http (ricadono nel caso ancora), né del tipo ftp (ricadono nel caso ancora ad oggetto). Il più comune è il caso di indirizzi malto, i quali vengono interpretati dal browser come un comando verso il programma gestore della posta elettronica, al fine di creare un nuovo messaggio che abbia come destinatario l'indirizzo specificato dopo i due punti.

Attributi:

- **Id**: numero seriale
- Indirizzo: indirizzo dell'oggetto destinazione
- Tipo: tipologia di indirizzo (valori possibili: mailto, news, telnet)

Generalizzaioni:

- Tag HTML

Blocco VBScript: si tratta di un frammento di codice, incorporato in un tag HTML di tipo <% interpretabile da un server ASP. Essendo fisicamente incorporato in un tag, è imprescindibile dalla pagina nella quale è situato, per cui è stato introdotto un numero seriale come identificatore.

Attributi:

- **Id**: numero seriale

Associazioni:

- Redirect con pagina web

Generalizzazioni:

- Tag HTML

Frameset: si tratta di un tag HTML presente solo in alcune particolari pagine web, nelle quali viene definita la suddivisione in frames della pagina.

Attributi.

- **Id:** numero seriale

Associazioni:

- Inclusione con frame (relazione di tipo composition)

Generalizzazioni:

- Tag HTML

Frame: quando una pagina web contiene una direttiva frameset, essa è suddivisa in più zone, chiamate frame. Un'ulteriore direttiva, individuata dal tag <FRAME indica al browser le caratteristiche delle singole zone.

Attributi:

- **Id**: numero seriale
- Nome: titolo del frame. Non è utilizzato come identificatore in quanto è possibile dare nomi uguali a frame di pagine diverse

Associazioni:

- Inclusione con frameset (relazione di tipo composition)
- Target, verso una pagina web

Generalizzazioni:

- Tag HTML

Applet: corrisponde al tag HTML nel quale è specificato il nome di un bytecode Java eseguibile da un browser dotato di Java Virtual Machine.

Attributi:

- **Id**: numero seriale

Associazioni:

- Esecuzione con Java class

Generalizzazioni:

- Tag HTML

Java class: si tratta dei bytecode java richiamati dal browser a seguito dell'interpretazione di un tag applet. Essi hanno sempre estensione .class

Attributi:

- **Id**: numero seriale
- Nomefile col quale si intende il percorso completo cui è possibile reperire il file

- Warning: indica che non si è riuscito ad associare all'immagine un nome di file consistente
- Nota: spazio di commento per l'utente

Associazioni:

- Esecuzione con applet

3.2 Associazioni

Builds, di tipo *association*, tra Pagina Client costruita (1..*) e Pagina Server (0..*): esprime il rapporto tra una Pagina Server in linguaggio Asp e la pagina client che risulta come risultato della sua esecuzione sulla macchina client.

Interfaccia, di tipo *association class*, tra Pagina Server (0..*) e Oggetto interfaccia (1..*): esprime il rapporto tra una pagina server e un oggetto che funge da interfaccia verso un database e che da essa sia stato creato.
Attributi:

- Linea: linea del file sorgente in cui è contenuta l'istruzione che istanzia un oggetto
- Nota: spazio di commento per l'utente

Redirect BJ, di tipo *association class*, tra Pagina web (0..*) e Blocco Javascript (0..*): esprime la possibilità che un blocco javascript possa redirigere il controllo verso una pagina web.
Attributi:

- Linea: linea del file sorgente in cui è contenuta l'istruzione di redirect
- Nota: spazio di commento per l'utente

Redirect MJ, di tipo *association class*, tra Pagina web (0..*) e Modulo Javascript (0..*): esprime la possibilità che un modulo javascript possa redirigere il controllo verso una pagina web.
Attributi:

- Linea: linea del file sorgente in cui è contenuta l'istruzione di redirect
- Nota: spazio di commento per l'utente

Redirect BV, di tipo *association class*, tra Pagina web (0..*) e Blocco VBScript (0..*): esprime la possibilità che un blocco VBScript possa redirigere il controllo verso una pagina web.
Attributi:

- Linea: linea del file sorgente in cui è contenuta l'istruzione di redirect

- Nota: spazio di commento per l'utente

Nota bene: se in un blocco o modulo, Javascript o VBScript, non figura alcun comando di redirect, si sottintende che, al termine dell'esecuzione, il controllo ritorna alla pagina che include tale blocco o modulo.

Inclusione, di tipo *aggregation*, tra Blocco Javascript (0..*) e modulo Javascript (0..*): esprime la possibilità che un blocco javascript inglobi in fase di esecuzione un modulo javascript, contenuto in un altro file.

Inclusione, di tipo *composition*, tra pagina web (1..*) e Tag HTML (1..*) : esprime semplicemente il fatto che i tag HTML sono compresi nelle pagine web e quindi ad esse indissolubilmente legati. D'altronde le pagine web devono contenere obbligatoriamente tag HTML.

Submit, , di tipo *association class*, tra form (0..1) e pagina web (0..*): esprime la posibilità che una pagina web possa essere richiamata da un form, in seguito all'operazione di submit. Se il form è di input/output, allora richiamerà una e una sola pagina web, altrimenti, se è un form di solo output, non parteciperà alla relazione.
Attributi:

- Metodo, che può prendere i valori GET (passaggio dei parametri sulla linea di comando, caso più semplice e veloce) e POST (passaggio dei parametri in un file allegato, caso più generale e sicuro)
- Nota: spazio di commento per l'utente

Link, di tipo *association*, tra ancora (1) e pagina web (0..*): esprime il collegamento ipertestuale unilaterale tra il tag HTML ancora e la pagina web da esso indicata.

Inclusione, di tipo *composition*, tra Frameset (0..*) e Frame (1): esprime il fatto che il tag frame che serve a definire una zona della pagina web e la pagina che vi deve essere caricata, sia compresa all'interno di un blocco delimitato dai tag <FRAMESET> e </FRAMESET>.

Target , di tipo association, tra Frame (1) e Pagina web (0..*): esprime il legame di collegamento ipertestuale tra il tag di definizione di un frame e la pagina web da esso indirizzata (si noti infatti che la cardinalità è la stessa della relazione link).

Esecuzione , di tipo association, tra Applet (1) e Java class (0..*): esprime il legame unilaterale tra il tag applet e il bytecode java da esso richiamato. La cardinalità 0 in Java class esprime il fatto che i bytecode java hanno, per convenzione, sempre estensione .class e potrebbero quindi essere individuati anche in assenza di questa relazione. In tal caso si possono fare due ipotesi: o l'oggetto non è collegato al resto dell'applicazione web, oppure è richiamato mediante interazioni non considerate (ad esempio da un altro Java class).

Visualizzazione, di tipo association, tra ancora ad immagine (1) e Immagine (0..*): esprime il legame unilaterale tra un tag che punti ad un file di tipo immagine (estensione .JPG o .GIF) e quest'ultimo file. La cardinalità 0 in Immagine è riservata alla possibilità che un'immagine non sia mai richiamata da alcun link. Anche in questo caso non è detto che ciò significhi che l'immagine sia estranea all'applicazione web.

Download, di tipo association, tra ancora ad oggetto (1) e Oggetto (1..*): esprime il legame unilaterale tra un tag applet che punta ad un file non di tipo immagine e quest'ultimo. Rientrano in questa categoria sia i richiami a file con estensione diversa da JPG, GIF, sia i collegamenti ipertestuali secondo il protocollo FTP.

3.3 Vincoli di integrità sui dati

- Un blocco javascript può essere incluso solo in un tag HTML incluso in una Pagina Client;
- Un blocco VBScript può essere incluso solo in un tag HTML incluso in una Pagina Server ASP;
- Il tipo di un oggetto interfaccia può essere non generico solo se si interfaccia con una pagina ASP;

- Una form di solo output non può essere in relazione submit con alcuna pagina web

4 Note sulla ristrutturazione del class diagram

4.1 Introduzione

La ristrutturazione del class diagram si prefigge alcuni scopi:

- eliminare attributi ridondanti;
- eliminare le gerarchie di generalizzazione;
- trovare gli identificatori primari.

4.1.1 Eliminazione degli attributi ridondanti

Consiste nell'esaminare tutti gli attributi di tutte le entità e Associazioni e verificare se alcuni di essi sono ridondanti o sono comunque ricavabili in funzione di altri attributi.

Per poter decidere la sorte di questi attributi è necessario uno studio, ancorché stimato, dei vantaggi e degli svantaggi che si possono avere in entrambi i casi.

In generale, la presenza di un attributo ridondante può velocizzare alcune particolari query; viceversa, la sua assenza alleggerisce l'occupazione di memoria e velocizza altre query tra cui sicuramente quelle di inserimento e aggiornamento.

4.1.2 Eliminazione delle gerarchie di generalizzazione

Le gerarchie di generalizzazione non sono un costrutto previsto dai database relazionali. E' sempre necessaria perciò una loro traduzione in classi e associazioni. Si presentano in generale tre alternative:

- unificare il padre della generalizzazione nei figli. In questo caso gli attributi e le associazioni relative al padre vengono duplicati per tutti i figli;
- unificare i figli della generalizzazione nel padre. In questo caso gli attributi di tutti i figli vengono aggiunti al padre (ammetteranno quindi valori nulli), così come le relazioni. Inoltre deve essere aggiunto nel padre un nuovo attributo che identifichi il tipo del figlio coinvolto in ogni tupla;
- introdurre un'ulteriore relazione, con cardinalità uno a uno dal padre verso ognuno dei figli.

4.1.3 Introduzione degli identificatori primari

Si tratta del primo passo nella ricerca delle chiavi primarie: riguarda in questo caso solo le classi e consiste nel trovare insiemi di attributi che godano della proprietà di chiave nell'ambito delle entità finora prese in considerazione.

Molte volte non è possibile trovare degli insiemi di attributi che soddisfino questa proprietà, per cui si introducono degli identificatori artificiali (che nel seguito verranno di solito indicati con Id) i quali identificano univocamente ogni tupla con un proprio numero seriale.

A volte si può scegliere di introdurre identificatori artificiali anche quando si vuole evitare di utilizzare complessi identificatori esterni.

4.2 Ristrutturazione del class diagram

In base a queste linee guida, si riporta ora un elenco delle attività di ristrutturazione del modello iniziale che portano alla successiva fase di traduzione verso il modello relazionale vero e proprio.

4.2.1 Eliminazione degli attributi ridondanti

Questa prima fase del lavoro risulta abbastanza semplice in quanto già nella prima fase si erano scelti una serie di attributi non ridondanti. Infatti, non essendoci ancora una precisa idea dei carichi in gioco, nonché delle particolari query richieste dal sistema, non si è ritenuto di dover aggiungere attributi al solo fine di ottenere una futura efficienza in alcune operazioni.

4.2.2 Eliminazione delle gerarchie di generalizzazione

Per eliminare le numerose gerarchie presenti nel modello si sono viceversa dovute applicare un po' tutte le possibilità previste: segue ora un elenco delle azioni intraprese.

- **Gerarchia tra Pagina HTML, Pagina Client costruita e Pagina Client**

Siccome Pagina Client non ha attributi né associazioni proprie, risulta naturale eliminarla, trasferendo quindi il suo legame gerarchico verso Pagina web ad entrambi i figli. Si noti che, così operando, bisogna ricordare che le operazioni che andrebbero

effettuate rispetto a Pagina Client, devono ora essere ripetute per ognuna delle sue entità figlie, e poi unificare i risultati.

- **Gerarchia tra Pagina ASP e Pagina Server**

La classe Pagina ASP non ha attributi né associazioni: risulta naturale quindi inglobarla nell'entità Pagina Server, aggiungendo ad essa un attributo tipo, uno dei cui valori possibili sia ASP.

- **Gerarchia tra Pagina Client costruita, pagina HTML, Pagina server e Pagina web**

In questo caso la classe padre Pagina web ha alcuni attributi propri e numerose associazioni, e ciò vale anche per le sue classi figlie. La soluzione più naturale risulta quindi quella di non eliminare alcuna classe ed introdurre delle relazioni ulteriori (che si potrebbero etichettare come di "inclusione") tra ognuna delle classi figlie e Pagina web. La cardinalità massima delle associazioni sarà ovviamente uno a uno, ma dal lato di Pagina web avremo ovviamente una cardinalità minima pari a 0.

- **Gerarchia tra Blocco javascript, form, applet, ancora ad immagine, ancora ad oggetto, frameset, ancora, ancora ad altro indirizzo, blocco VBScript e Tag HTML**

Si ha a che fare ora con una gerarchia tra molte classi figlie, abbastanza diverse tra loro, e una classe padre: si scarta quindi subito l'idea di accorpare i figli nel padre. La molteplicità dei figli porta ad escludere anche la possibilità di introdurre una relazione uno a uno tra il padre, Tag HTML, e ognuno di essi. Per esclusione, si decide quindi di eliminare la classe padre. Questa classe ha due attributi, Linea e Nota, che verranno replicati nelle classi figlie e una associazione, Inclusione, verso Pagina web che sarà anch'essa replicata.

Le unificazioni fatte portano anche ad una rettifica di un'associazione: non ha più senso infatti considerare un blocco VBScript semplicemente come specializzazione di un Tag HTML (col vincolo che tale tag debba trovarsi in una pagina web di tipo server e asp), ma può essere considerato incluso direttamente in una pagina server. In altre parole la relazione di inclusione tra Pagina web e Blocco VBScript si sposta tra

Pagina Server e Blocco VBScript. Inoltre viene modificato anche il vincolo d'integrità sui dati: un Blocco VBScript può essere in relazione con una Pagina Server se e solo se quest'ultima è di tipo ASP.

Situazione analoga si ha per le pagine Client contenenti Blocchi Javascript. La relazione di inclusione, oltre a essere spostata, verrà anche duplicata, vista la ristrutturazione fatta per Pagina Client.

4.2.3 Introduzione degli identificatori primari

Per quanto riguarda gli identificatori principali, vengono confermati tutti quelli introdotti in fase di descrizione del modello concettuale.
Ricapitolando, alcune classi hanno come identificatore principale un numero seriale, precisamente:

- Pagina Client costruita
- Oggetto interfaccia
- Blocco javascript
- Pagina web
- Frame
- Blocco VBScript
- Frameset
- Ancora
- Form
- Parametri Form
- Applet
- Ancora ad immagine
- Ancora ad oggetto
- Ancora ad altro indirizzo
- Java class
- Immagine
- Oggetto
- Modulo Javascript

Per le altre classi si è scelto come identificatore l'attributo Nomefile. Affinché tale attributo possa rispettare la condizione di unicità, è necessario che includa anche il

path e che sia rilevato, ove possibile, in maniera univoca (assoluta, oppure relativa ad un riferimento noto).

4.3 Class diagram ristrutturato

Nella figura seguente viene riportato il diagramma del modello entità-relazione ristrutturato secondo quanto detto nel precedente paragrafo.

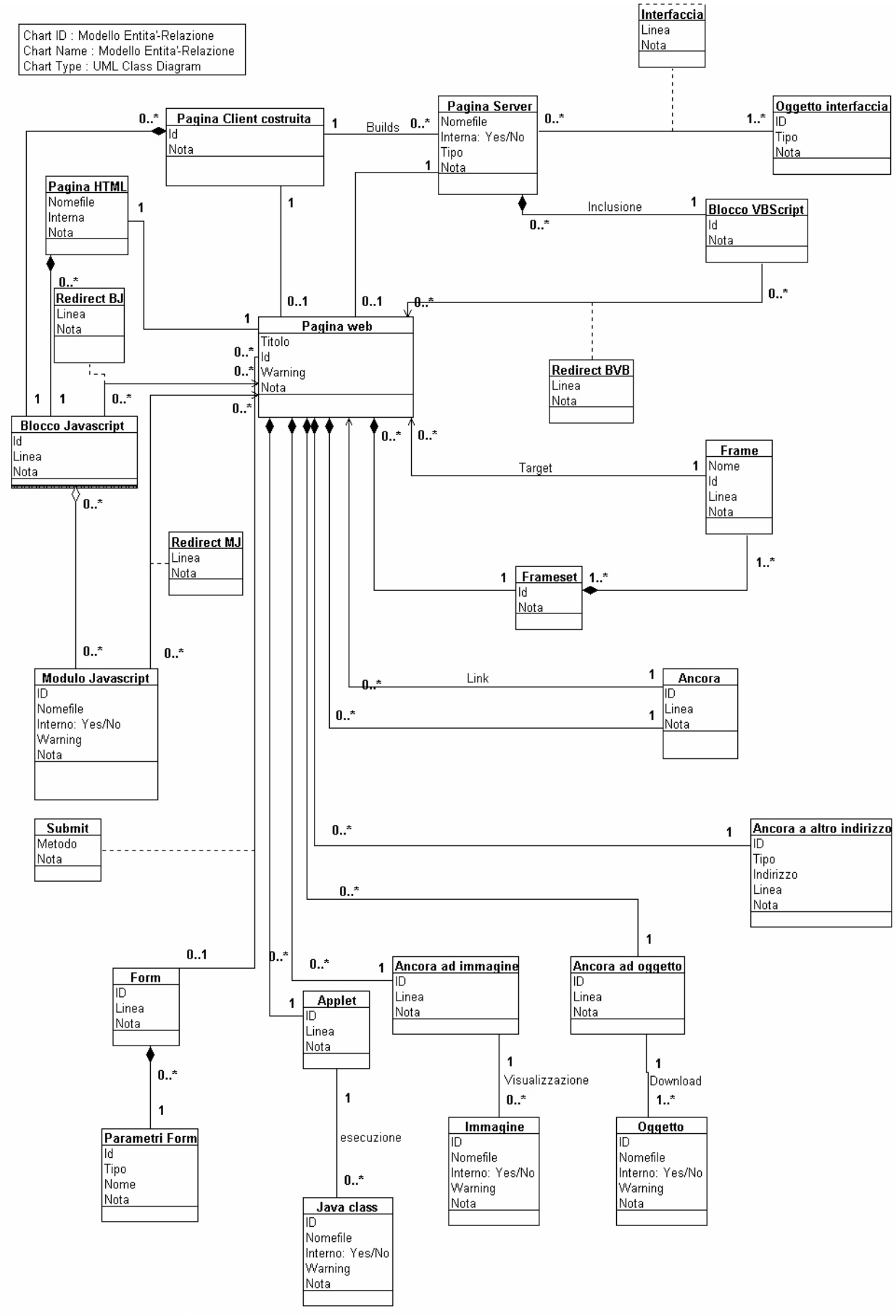


Figura 2 – Class diagram ristrutturato

5 Il modello relazionale

Questa fase della progettazione del database prende come input il class diagram risultante dopo la ristrutturazione. La fase successiva nello sviluppo della base di dati è la traduzione verso un modello implementativo. Il modello maggiormente utilizzato al giorno d'oggi è quello dei database relazionali.
In tali database il componente di base è la cosiddetta relazione o tabella. Il significato di tale termine in questo contesto è assai semplice: si tratta di una tabella, le cui colonne sono dette campi e rappresentano, nel loro complesso, lo schema della tabella, e le cui righe rappresentano ognuna un record della tabella.
Nella definizione di uno schema relazionale bisogna poi indicare i vincoli tra i dati. Distinguiamo tre tipologie di vincoli:

- Vincoli sui dati, ovvero riguardanti singoli valori (vincoli di dominio)
- Vincoli intrarelazionali, ovvero verificabili nell'ambito di un record di una tabella. Tra essi c'è il vincolo di chiave primaria;
- Vincoli interrelazionali, che riguardano più tabelle contemporaneamente. Tra essi ci sono i vincoli di chiave esterna e di integrità referenziale.

Elenchiamo ora una serie di strategie generali di traduzioni verso il modello relazionale.

5.1 Classi isolate

Si tratta del caso più semplice: la classe si traduce in una tabella che ha come chiave primaria l'identificatore primario della classe stessa.

5.2 Classi in associazione molti a molti

Si premette che, con questa dizione, ci si riferisce sempre alle cardinalità massime dell'associazione. Come si vedrà, le cardinalità minime influiranno anch'esse.
La traduzione naturale in tali casi consiste in tre tabelle: due per le classi recanti gli attributi di queste (chiave primaria rimane l'identificatore primario) e una terza che rappresenta l'associazione, che ha gli attributi dell'associazione stessa, più gli attributi chiave delle classi partecipanti i quali, nel loro complesso, sono la chiave primaria per questa tabella.

Esistono ovviamente dei vincoli di integrità referenziale tra gli attributi chiave delle tabelle risultanti dalle classi e gli attributi omonimi nelle tabelle risultanti da associazioni.
Tale strategia si può generalizzare anche al caso di associazioni con più di due entità.
Al fine di rendere più chiaro il modello relazionale, si effettuano di solito alcune ridenominazioni. Tipicamente vengono ridenominati gli attributi che vengono coinvolti in una tabella che traduce una relazione, con un nome che ricordi la classe di provenienza di tali attributi.

5.3 Classi in associazioni uno a molti

In questo caso è ancora possibile la traduzione in tre tabelle vista per le associazioni molti a molti, e ha anche il vantaggio di non avere valori nulli in alcun campo chiave anche in presenza di cardinalità minime nulle. Si può però adottare una soluzione più semplice ed economica: la classe con cardinalità uno e l'associazione vengono tradotte in un'unica tabella, contenente gli attributi della classe, dell'associazione, e la chiave dell'altra classe (che viene invece tradotta nel modo naturale). Chiave per la prima tabella rimane quella della classe che vi partecipa, mentre vige un vincolo di integrità referenziale tra gli attributi chiave della seconda tabella e i loro omonimi nella prima.

5.4 Classi in associazione in relazione uno a uno

In questo caso ci sono tre diverse possibilità di traduzione e la scelta tra esse si basa sulla cardinalità minima delle entità:

- Se entrambe le classi non sono opzionali nell'associazione si possono tradurre in due modi equivalenti con due tabelle, nel modo visto per le relazioni uno a molti;
- Se una delle due classi è opzionale, allora è possibile la traduzione in due tabelle a patto di tradurre la classe opzionale in una tabella intera (infatti la sua chiave primaria non può esserlo anche per la relazione);
- Se entrambe le classi sono opzionali, l'unica traduzione possibile è quella in tre tabelle vista per le associazioni molti a molti.

Si è cercato di applicare queste regole di traduzione al class diagram trovato, perseguendo, ove possibile, l'obiettivo di ridurre per quanto possibile il numero di

tabelle (ciò significa anche ridurre la lunghezza e i cammini di join). Per questo motivo si riporta direttamente la descrizione, completa e puntuale, delle tabelle del database relazionale.

6 Descrizione dello schema relazionale

6.1 Introduzione

In questa sezione vengono descritti gli schemi delle tabelle che devono essere state realizzate, gli attributi e i vincoli di integrità referenziale che legano gli attributi di due o più tabelle.
La descrizione viene fatta secondo lo schema seguente:

Nome Tabella

- *Attributo 1*
  - *Tipo*
  - *(Richiesto, non richiesto)*
  - *(duplicati ammessi, non ammessi)*

...

- *Attributo N*
  - *Tipo (Richiesto, non richiesto)*
  - *(duplicati ammessi, non ammessi)*

Vincoli di integrità

- *Attributo x con Nometabella.Attributoy*

...

L'attributo o gli attributi che sono chiave primaria vengono sottolineati.
Le tabelle del Database sono state realizzate con Microsoft Access e si rimanda al file per una visualizzazione completa.

6.2 Descrizione delle tabelle

Ancora

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No

- IdPaginaWeb
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- IdPaginaWebLinkata
  - Tipo:Intero
  - Richiesto:Si
  - Duplicati: Si
- Linea
  - Tipo:Intero
  - Richiesto:No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- IdPaginaWebLInkata con **PaginaWeb**.ID

Ancora ad altro indirizzo

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No
- Indirizzo
  - Tipo: Testo
  - Richiesto:Si
  - Duplicati: Si
- IdPaginaWeb
  - Tipo:Intero
  - Richiesto:Si

        - Duplicati: Si
    - Linea
        - Tipo:Intero
        - Richiesto:No
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID

Ancora ad immagine

- ID
    - Tipo: Contatore
    - Richiesto:Si
    - Duplicati: No
- IdPaginaWeb
    - Tipo: Intero
    - Richiesto:Si
    - Duplicati: Si
- Immagine
    - Tipo:Intero
    - Richiesto:Si
    - Duplicati: Si
- Linea
    - Tipo:Intero
    - Richiesto:No
    - Duplicati: Si
- Nota
    - Tipo:Memo
    - Richiesto:No
    - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- Immagine con **Immagine**.ID

Ancora ad oggetto

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No
- IdPaginaWeb
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- Oggetto
  - Tipo:Intero
  - Richiesto:Si
  - Duplicati: Si
- Linea
  - Tipo:Intero
  - Richiesto:No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- Oggetto con **Oggetto**.ID

Applet

- ID
  - Tipo: Contatore

        - Richiesto:Si
        - Duplicati: No
    - IdPaginaWeb
        - Tipo: Intero
        - Richiesto:Si
        - Duplicati: Si
    - Class
        - Tipo:Intero
        - Richiesto:Si
        - Duplicati: Si
    - Linea
        - Tipo:Intero
        - Richiesto:No
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- Class con **Java Class**.ID

Blocco JavaScript

- ID
    - Tipo: Contatore
    - Richiesto:Si
    - Duplicati: No
- IdPaginaWeb
    - Tipo:intero
    - Richiesto:Si
    - Duplicati: Si
- Linea
    - Tipo:Intero

- Richiesto:No
- Duplicati: Si

- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.Id

Blocco VBScript

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No
- PaginaServer
  - Tipo: Testo
  - Richiesto:Si
  - Duplicati: No
- Linea
  - Tipo:Intero
  - Richiesto:No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- PaginaServer con **PaginaServer**.Nomefile

Form

- ID

- Tipo: Contatore
- Richiesto:Si
- Duplicati: No

- IdPaginaWeb
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- Linea
  - Tipo:Intero
  - Richiesto:No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID

Frame

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No
- Nome
  - Tipo: Testo
  - Richiesto:No
  - Duplicati: Si
- IdSource
  - Tipo: Intero
  - Richiesto:No
  - Duplicati: Si
- IdFrameset
  - Tipo: Intero

- Richiesto:Si
- Duplicati: Si

- Linea
  - Tipo:Intero
  - Richiesto:No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdSource con **PaginaWeb**.ID
- IdFrameset con **Frameset**.ID

Frameset

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No
- IdPaginaWeb
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- Linea
  - Tipo:Intero
  - Richiesto:No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID

Immagine

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No
- Nomefile
  - Tipo: Testo
  - Richiesto:Si
  - Duplicati: No
- Interno
  - Tipo: (Si,No)
  - Richiesto:No
  - Duplicati: Si
- Warning
  - Tipo: Testo
  - Richiesto: No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

InclusioneModuloJavascript

- BloccoJavascript
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- ModuloJavascript
  - Tipo: Testo

- Richiesto:Si
- Duplicati: Si

Vincoli di Integrità

- BloccoJavascript con **Blocco Javascript**.ID
- Modulo Javascript con **Modulo Javascript**.ID

Java Class

- ID
  - Tipo: Contatore
  - Richiesto:Si
  - Duplicati: No
- Nomefile
  - Tipo: Testo
  - Richiesto:Si
  - Duplicati: No
- Interno
  - Tipo: (Si,No)
  - Richiesto:No
  - Duplicati: Si
- Warning
  - Tipo: Testo
  - Richiesto: No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Modulo Javascript

- ID
  - Tipo: Contatore
  - Richiesto:Si

        - Duplicati: No
    - Nomefile
        - Tipo: Testo
        - Richiesto:Si
        - Duplicati: No
    - Interno
        - Tipo: (Si,No)
        - Richiesto:No
        - Duplicati: Si
    - Warning
        - Tipo: Testo
        - Richiesto: No
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Oggetto

- ID
    - Tipo: Contatore
    - Richiesto:Si
    - Duplicati: No
- Nomefile
    - Tipo: Testo
    - Richiesto:Si
    - Duplicati: No
- Interno
    - Tipo: (Si,No)
    - Richiesto:No
    - Duplicati: Si
- Warning
    - Tipo: Testo
    - Richiesto: No

        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Oggetto Interfaccia

- ID
    - Tipo: Contatore
    - Richiesto:Si
    - Duplicati: No
- PaginaServer
    - Tipo: Testo
    - Richiesto:Si
    - Duplicati: Si
- Tipo
    - Tipo: Testo
    - Richiesto:No
    - Duplicati: Si
- Nota
    - Tipo:Memo
    - Richiesto:No
    - Duplicati: Si
- Linea
    - Tipo:intero
    - Richiesto:No
    - Duplicati: Si

Vincoli di Integrità

- PaginaServer con **PaginaServer**.Nomefile

Pagina Client Costruita

- ID
    - Tipo: Contatore
    - Richiesto:Si

        - Duplicati: No
    - PaginaServer
        - Tipo: Testo
        - Richiesto: Si
        - Duplicati: Si
    - IdPaginaWeb
        - Tipo: Intero
        - Richiesto:Si
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Vincoli di Integrità

- PaginaServer con **PaginaServer**.Nomefile
- IdPaginaWeb con **PaginaWeb**.ID

Pagina HTML

- Nomefile
    - Tipo: Testo
    - Richiesto:Si
    - Duplicati: No
- Interna
    - Tipo: (Si,No)
    - Richiesto: No
    - Duplicati: Si
- IdPaginaWeb
    - Tipo: Intero
    - Richiesto:Si
    - Duplicati: Si
- Nota
    - Tipo:Memo
    - Richiesto:No

- Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID

Pagina Server

- Nomefile
  - Tipo: Testo
  - Richiesto:Si
  - Duplicati: No
- Interna
  - Tipo: (Si,No)
  - Richiesto: No
  - Duplicati: Si
- Tipo
  - Tipo: Testo
  - Richiesto: No
  - Duplicati: Si
- IdPaginaWeb
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID

Pagina Web

- ID
  - Tipo: Contatore
  - Richiesto:Si

        - Duplicati: No
    - Titolo
        - Tipo: Testo
        - Richiesto: No
        - Duplicati: Si
    - Warning
        - Tipo: Testo
        - Richiesto: No
        - Duplicati: Si
    - Nome
        - Tipo: Testo
        - Richiesto: No
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Parametri Form

- ID
    - Tipo: Contatore
    - Richiesto:Si
    - Duplicati: No
- Tipo
    - Tipo: Testo
    - Richiesto: No
    - Duplicati: Si
- Nome
    - Tipo: Testo
    - Richiesto: No
    - Duplicati: Si
- IdForm
    - Tipo: Intero

        - Richiesto:Si
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Vincoli di Integrità

- IdForm con **Form**.ID

Redirect BJ

- Id
    - Tipo: Intero
    - Richiesto:Si
    - Duplicati: No
- IdPaginaWeb
    - Tipo: Intero
    - Richiesto:Si
    - Duplicati: Si
- IdBloccoJavascript
    - Tipo: Intero
    - Richiesto:Si
    - Duplicati: Si
- Linea
    - Tipo:Intero
    - Richiesto:No
    - Duplicati: Si
- Nota
    - Tipo:Memo
    - Richiesto:No
    - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- IdBloccoJavascript con **BloccoJavascript**.ID

Redirect BV

- Id
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: No
- IdPaginaWeb
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- IdBloccoVBscript
  - Tipo: Intero
  - Richiesto:Si
  - Duplicati: Si
- Linea
  - Tipo:Intero
  - Richiesto:No
  - Duplicati: Si
- Nota
  - Tipo:Memo
  - Richiesto:No
  - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- IdBloccoVBscript con **BloccoVBscript**.ID

Redirect MJ

- Id
  - Tipo: Intero

        - Richiesto:Si
        - Duplicati: Si

    - IdPaginaWeb
        - Tipo: Intero
        - Richiesto:Si
        - Duplicati: Si
    - ModuloJavascript
        - Tipo: Testo
        - Richiesto:Si
        - Duplicati: Si
    - Linea
        - Tipo:Intero
        - Richiesto:No
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- ModuloJavascript con **ModuloJavascript**.Nomefile

Submit

- IdPaginaWeb
    - Tipo: Intero
    - Richiesto:Si
    - Duplicati: Si
- IdForm
    - Tipo: Intero
    - Richiesto:Si
    - Duplicati: Si
- Metodo

        - Tipo: (GET,POST)
        - Richiesto: No
        - Duplicati: Si
    - Nota
        - Tipo:Memo
        - Richiesto:No
        - Duplicati: Si

Vincoli di Integrità

- IdPaginaWeb con **PaginaWeb**.ID
- IdForm con **Form**.ID

CAPITOLO 9 - IL FILE PRODOTTO DALL'ESTRATTORE

1 Introduzione

In questo capitolo si specificheranno le informazioni che bisogna estrarre dai file sorgente e la forma di rappresentazione con cui saranno memorizzate in un apposito file.

Più precisamente si vogliono definire:

- la sintassi dei tag e comandi nei file sorgenti che si debbono riconoscere;
- il comportamento che deve avere l'estrattore quando incontra tali informazioni nel file sorgente;
- la sintassi dei tag del file risultante in cui registrare le informazioni estratte.

Ciò, inoltre, permetterà di meglio definire i requisiti del tool astrattore che è stato definito nell'architettura complessiva del sistema.

La scelta della sintassi del file intermedio risente di alcune scelte di massima riguardanti i compiti dell'estrattore e dell'astrattore, che si devono anticipare in questa sede.

Si è scelto di seguire una precisa strategia che renda vere le seguenti asserzioni:

- L'estrattore (parser) è senza stato;
- Il popolamento del database è a carico dell'astrattore successivo.

La prima affermazione comporta alcune importanti conseguenze. Prima di tutto il parser può operare in un unico passo, e questo rappresenta una notevole semplificazione nell'implementazione. A conferma di quest'affermazione, si può vedere la figura che esplica l'associazione tra le informazioni provenienti dai file sorgenti e i tag generati sui file intermedi: ogni tag è generato da uno e un solo punto d'informazione, quindi non c'è necessità di mantenere uno stato tra l'analisi di diversi punti d'informazione.

Si è definito in questo contesto come punto d'informazione una qualsiasi operazione tra analisi di un tag HTML rilevante, analisi di un metodo ASP rilevante, apertura e chiusura di file appartenenti all'applicazione web in analisi.

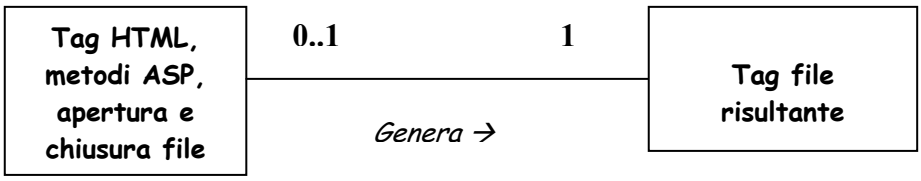


Figura 1 – Legame grafico tra i tag nel codice sorgente e i tag nel file risultante dall'estrattore

Il compito dell'estrattore è dunque quello di filtrare le informazioni utili tra tutte quelle rintracciabili nell'applicazione web e riprodurle, secondo la sintassi assegnata, nel file intermedio.
All'astrattore viene demandata l'analisi del file con la rappresentazione intermedia e l'inserimento conseguente delle informazioni trovate nel database.

Seguendo questa strategia, si ottiene una sintassi del file di scambio che assomiglia più ad un filtraggio del contenuto dei file sorgente, piuttosto che ad una rappresentazione dei dati da inserire nel database. E' da tener conto che figureranno in esso anche dei tag risultanti da azioni compiute dal parser (apertura, chiusura file).
Nei prossimi paragrafi verrà prima data una descrizione sintetica della sintassi di HTML, Javascript e VBScript, orientata solamente ai tag e comandi contenenti le informazioni che si vogliono catturare, poi una descrizione della sintassi delle informazioni nel file risultante.

2 Sintassi dei linguaggi sorgenti

In questo paragrafo verrà descritta, in maniera essenziale, la sintassi dei linguaggi da analizzare, limitatamente solo alle informazioni da ricavare.

2.1 HTML

Questo linguaggio è organizzato in tag, ovvero in strutture sintattiche nella forma
<NOMETAG parametri >
zona sotto l'influenza del tag NOMETAG
</NOMETAG>

In quella che è stata indicata come zona di influenza del tag, può comparire qualsiasi cosa, compresi altri tag.
Ciò premesso, ecco un elenco dei tag da esaminare:

Tag TITLE
Sintassi:
<TITLE>
titolo
</TITLE>

Questo tag permette di associare un titolo ad una pagina web. Il titolo consiste nel testo compreso tra i due tag delimitatori. Tra essi non vi possono essere altri tag HTML. Possono però esserci tag di un altro linguaggio, ad esempio ASP. In tal caso non è più possibile conoscere in maniera certa in fase di analisi statica il titolo della pagina, per cui bisogna accontentarsi di segnalarne la variabilità.

Tag FRAMESET
Sintassi:
<FRAMESET [parametri]>
…
</FRAMESET>

Questo tag definisce una suddivisione, verticale o orizzontale, dello schermo in due zone, ad ognuna delle quali può essere contenitore di una diversa pagina web.
Per permettere una definizione di più di due zone, è possibile nidificare tag frameset. La zona d'influenza del tag frameset è anche l'unica nella quale sia possibile trovare il tag frame. Nessun parametro del tag frameset rientra tra le informazioni da ricavare.

Tag FRAME

Sintassi:

<FRAME SRC="pagina contenuta" [NAME="nome"] [altri parametri]>

Questo tag contiene la definizione degli elementi contenuti nei frame introdotti dal tag frameset. Il parametro src indica il nome della pagina che deve essere contenuta nel frame in questione, il parametro name serve invece ad indicarne una specie di titolo, cui si potrà fare riferimento con altri tag.

Tag SCRIPT

Sintassi:

<SCRIPT [language="linguaggio"] [source="nome del file modulo"] [altri parametri]>

…

</SCRIPT>

Questo tag ha lo scopo di includere all'interno della pagina un frammento di codice in un linguaggio di scripting. Il parametro language contiene il nome del linguaggio di scripting, qualora sia omesso, deve considerarsi Javascript. Il parametro source indica se il frammento di codice si trova su di un altro file, interamente scritto in questo linguaggio di scripting. Qualora tale parametro sia omesso, si intende che il frammento di codice sia presente nella zona d'influenza del tag script.

Tag <%

Sintassi:

<% [language="linguaggio di script"]

codice linguaggio di scripting

%>

Questo tag introduce un frammento di codice in un linguaggio di scripting lato server, come VBScript o Jscript, ad esempio. Il frammento di codice costituisce la zona d'influenza del tag. Se il parametro language è omesso, il valore di default è VBScript.

Tag BODY

Sintassi:

<BODY>

corpo del documento

</BODY>

Questo tag delimita il corpo del documento, ovvero la parte di esso contenente l'ipertesto che viene interpretato dal browser. L'importanza di questo tag deriva dal fatto che, dal punto in cui si incontra questo tag, ogni testo al di fuori del simbolo di tag, va interpretato come stringa da visualizzare.

Tag FORM

Sintassi:

<FORM [METHOD=GET|POST] [ACTION=' 'Nome del file cui vengono inviati i dati'] [altri parametri]>

zona d'influenza

</FORM>

Questo tag definisce un form, ovvero un meccanismo d'interazione tra l'utente e il sistema. Tra i parametri, action definisce l'applicazione che usufruirà dei dati ricavati in input dalla form, method il modo con cui tali dati vengono passati (get: dati passati sulla linea di comando, post: dati impacchettati in un file e inviati, eventualmente codificati). La zona d'influenza del tag form comprende sempre dei tag atti alla definizione dei meccanismi di input utilizzati (vedi tag input, textarea, img nel seguito).

Tag INPUT

Sintassi:

<INPUT TYPE=tipo [NAME=name] […] >

Questo tag serve a definire punti di interazione tra l'utente e il sistema. Il parametro tipo ammette i seguenti valori:

- **CHECKBOX**, crea delle caselle che assumono un valore booleano a seconda che siano o meno selezionate;
- **HIDDEN**, non viene visualizzato sul video, ma viene comunque elaborato dal server quando il form viene inviato dall'utente: serve in pratica a fornire dei parametri all'applicazione server;
- **IMAGE**, consente di definire un'immagine come equivalente al pulsante submit. Quando l'utente clicca sopra l'immagine, il form viene regolarmente inviato al server;
- **PASSWORD**, permette d'inserire una password che non apparirà sullo schermo. Generalmente, per ogni carattere immesso dalla tastiera viene visualizzato un asterisco;
- **RADIO**, definisce delle caselle che possono essere selezionate in modo mutuamente esclusivo. Viene molto utilizzato nei questionari dove è richiesta una sola scelta fra le tante messe a disposizione;
- **RESET**, crea un pulsante che, se premuto, azzera completamente i dati immessi fino a quel momento e visualizza nuovamente il form com'era in origine (situazione di default);
- **SUBMIT**, crea un pulsante che, se premuto, invia al server i dati contenuti nel form, in maniera tale da poter essere elaborati. Per l'utente rappresenta il pulsante su cui cliccare per confermare l'invio del form;
- **TEXT**, crea un campo nel quale è possibile inserire una sola linea di testo.

Il parametro name può fungere da riferimento per un blocco javascript o per l'applicazione server, per potersi riferire ad uno in particolare tra i valori ottenuti.

La presenza di questi tag al di fuori della zona d'influenza del tag form, sta a significare, in generale, che si tratti di strutture destinate al solo output.

Tag SELECT

Sintassi:

<SELECT [NAME=nome] [altri parametri]>
zona d'influenza
</SELECT>

Questo tag appartiene alla categoria dei parametri di un form, definita in precedenza. Tale parametro però ha una struttura più complessa, per cui ha bisogno di un suo tag esclusivo. Nella zona d'influenza devono esserci i possibili valori che può assumere il campo di selezione, identificati dal tag <OPTION>

Tag TEXTAREA

Sintassi:
<TEXTAREA [NAME=nome] [altri parametri]>
zona d'influenza
</TEXTAREA>

Anche questo tag è utilizzabile da un form, e serve ad interagire con una casella di testo a più righe.

Tag APPLET

Sintassi:
<APPLET CODE=' 'Nome del file che contiene il bytecode della classe java' [altri parametri]>
zona d'influenza
</APPLET>

Questo tag, introdotto solo nelle versioni più recenti di HTML, serve ad innescare nel browser l'esecuzione di un'applet Java. Tra i parametri figura ovviamente il percorso del file java, che deve essere nel formato interpretabile dalla Java Virtual Machine implementata all'interno del browser, detto bytecode. Nella zona d'influenza possono trovarsi tra l'altro tutti i parametri di input dell'applet, all'interno di appositi tag <PARAM>

Tag IMG

Sintassi:

<IMG SRC="nome del file immagine" [altri parametri]>

Questo tag serve ad inserire un'immagine in una pagina web. Tale immagine deve essere in uno dei formati immediatamente riconoscibili dal browser, tra cui ci sono sicuramente BMP, GIF e JPEG. Tra i parametri c'è l'indirizzo dell'immagine è il parametro SRC.

Tag A

Sintassi:

<A HREF="indirizzo cui punta il collegamento" […] >

zona d'influenza

</A>

Questo tag, universalmente conosciuto come tag ancora, permette di definire ogni tipo di link ipertestuale. Il parametro indirizzo può rendere diversi significati all'ancora, a seconda della sua tipologia. Infatti l'indirizzo può individuare:

- Una pagina sullo stesso web della pagina chiamante (in tal caso l'indirizzo è espresso in maniera relativa rispetto alla directory sorgente dell'applicazione web);
- Una pagina di un altro web (in tal caso l'indirizzo è espresso in modo globale, secondo il protocollo http);
- Un'altra zona della stessa pagina (in tal caso l'indirizzo è espresso nella forma nome_della_pagina#nome_della_zona;
- Un file di cui si può effettuare il download (ovvero in un formato diverso da quelli interpretabili dal browser oppure indirizzato secondo il protocollo FTP), interno o esterno al sito web;
- Un indirizzo di posta (protocollo mailto) oppure un altro indirizzo (protocolli news, telnet.

In tutti questi casi la zona sensibile del link sarà costituita da tutti gli oggetti visualizzati presenti nella zona d'influenza del tag.

2.2 Javascript

Il linguaggio Javascript ha una sintassi abbastanza somigliante a quella del linguaggio C++. Si hanno quindi chiamate di metodi di classi nella forma *classe.metodo*. Si è già visto, in sede di HTML, come iniziano e finiscono le zone di linguaggio Javascript in una pagina web. Un solo comando javascript è stato analizzato in questa sede.

Comando location.href

Sintassi:

location.href ("indirizzo della pagina destinazione")

Questo comando permette di effettuare, all'interno di un blocco Javascript, il redirect verso un'altra pagina web. L'importanza di questo comando è data dalla possibilità di spostarsi tra le pagine di testo senza l'interazione dell'utente.

2.3 VBScript

Il linguaggio VBScript ha una sintassi somigliante a quella del Visual Basic. In questa sede si vogliono esaminare in particolare le classi aggiunte a VBScript dalla tecnologia ASP, per permettere interazioni con le pagine web. I comandi che si esaminano sono i tre seguenti:

Comando response.write

Sintassi:

response.write "stringa da visualizzare"

Questo comando serve ad aggiungere una riga di testo nella pagina client costruita come output di un'esecuzione della pagina server. L'importanza di questo tag dipende dal fatto che dà la certezza del fatto che la pagina server in questione generi una pagina client.

Comando response.redirect

Sintassi:

response.redirect "indirizzo della pagina web destinazione"

Questo comando serve a ridigere l'esecuzione verso un'altra pagina web, server o client, analogamente a location.href in Javascript. Anche in questo caso l'indirizzo può riferirsi ad una pagina interna o esterna all'applicazione web, ad una pagina attiva o passiva.

Comando server.createobject

Sintassi:

server.createobject "oggetto interfaccia"

Questo comando serve a creare un meccanismo di interazione tra una pagina server ed un oggetto, come un database oppure un messaggio di posta elettronica, tramite un'interfaccia visibile dal linguaggio. Tra le possibili interfacce previste da ASP, si possono citare ADODB.Connection e ADODB.Recordset per l'interazione con i database, CDONTS.Newmail per l'interazione con i messaggi di posta elettronica.

3 Sintassi e semantica del file risultante dall'estrattore

3.1 Convenzioni

E' riportato di seguito il formato con cui le informazioni estratte sono registrate sul file costituente la forma di rappresentazione intermedia dell'applicazione analizzata.
Per ciascun file sorgente analizzato si ha una corrispondente forma intermedia. Questa è costituita da una serie di tag, ciascuno dei quali corrispondente ad uno degli elementi, prima descritti, riconosciuti nel file sorgente.
La sequenza con cui i tag sono presenti nella forma intermedia è quella lessicografica con cui i corrispondenti elementi sono presenti nel file sorgente.

<Nometag>
[<Attributo1=*Valore*>]
[<Attributo2=*Valore*>]
[…]
[<AttributoN=*Valore*>]
[</Nometag>]

Modalità di estrazione delle informazioni
Semantica

Nello schema precedente:

- Nometag indica l'identificativo del tag del file risultante che si sta descrivendo;
- Attributo*n* indica l'identificativo dell'n-esimo attributo relativo al tag in esame;
- […] sta ad indicare gli eventuali altri attributi opzionali
- per modalità di estrazione delle informazioni si intende l'insieme delle regole che devono essere seguite dal tool estrattore per ricavare dal sorgente le informazioni richieste nel tag del file obiettivo;

- per semantica si è intesa l'interpretazione che deve dare al tag in questione del file obiettivo un tool che abbia come obiettivo l'inserimento delle informazioni nel database definito nel capitolo precedente.

3.2 Descrizione dei tag della forma di rappresentazione intermedia

In questo paragrafo verranno elencati tutti i tag previsti dalla sintassi del file risultante dall'estrattore.
Si premette in questa sede che il parametro LINEA presente in molti dei tag corrisponde all'informazione sul numero di linea del file sorgente nel quale incomincia il tag o comando da cui scaturisce il tag del file risultante.

Tag APERTURA

<APERTURA>
    <NOMEFILE='*nomefile analizzato con percorso ed estensione*'>
</APERTURA>

Questo tag viene inserito nel file ogni volta che il parser apre un nuovo file da analizzare. L'astrattore, poi, dovrebbe creare nel database un nuovo record di una e una sola delle seguenti tabelle a seconda dell'estensione del file analizzato:

- "Pagina Web" (estensioni .htm, .html, .asp)
- "Modulo Javascript" (estensione .js)
- "Javaclass" (estensione .class)
- "Immagine" (estensioni .gif, .jpg)
- "Oggetto" (qualsiasi altra estensione).

L'attributo "interno" sarà sempre pari a vero.
Se viene creato un record della tabella "Pagina Web" allora deve essere creato anche un record di una delle seguenti tabelle:

- "Pagina Server" (estensione .asp)
- "Pagina HTML" (estensioni .htm, .html)

Per quanto riguarda nomefile, esso viene fornito dal parser con tutto il percorso, specificato in maniera assoluta o relativa. Sarà compito dell'astrattore ricavare il percorso esatto del file a partire da un riferimento relativo fissato rispetto alla radice dell'applicazione web analizzata, che è stata specificata dall'utente.

Tag CHIUSURA

<CHIUSURA>

Questo tag corrisponde alla condizione che il file che si è aperto per l'analisi è giunto alla fine. Questo tag si applica a tutte le tipologie di file visti per il tag apertura. Nel caso di file di tipo Immagine, Oggetto o Javaclass, il tag CHIUSURA verrà a trovarsi immediatamente dopo il tag APERTURA.

Tag TITOLO

<TITOLO>
    <TITOLO='*titolo della pagina*'>
</TITOLO>

Questo tag scaturisce direttamente dal tag TITLE di HTML, che si può trovare sia in una pagina Client che in una pagina Server.

Quando l'astrattore incontra tale tag, sa che si riferisce al file attualmente in analisi, quindi dovrà effettuare una query di aggiornamento al database per settare il titolo della pagina, che precedentemente era stato lasciato non specificato.

Tag FRAMESET

<FRAMESET>
    <LINEA=*numero di linea*>
</FRAMESET>

Questo tag corrisponde all'omonimo tag HTML, utilizzato per definire una suddivisione in tabelle della pagina web.

L'astrattore dovrà semplicemente inserire una nuova istanza alla tabella Frameset, indicando la linea e la pagina web di appartenenza del tag (ovvero quella in analisi).

Tag FRAME

<FRAME>

<LINEA=*numero di linea*>

<SOURCE=*'pagina sorgente'*>

[<NOME=*'nome del frame'*>]

</FRAME>

Il tag FRAME corrisponde all'omonimo tag di HTML. Tra i parametri di tale tag, il parser riconosce solo l'indirizzo della pagina che si posizionerà nel frame e, eventualmente, il nome del frame (si tratta di un'etichetta). L'indirizzo della pagina sorgente viene restituito dal parser senza alcun controllo: potrà quindi trattarsi anche di un'espressione valutabile in fase di esecuzione, quindi non risolvibile staticamente.

L'astrattore dovrà occuparsi dell'interpretazione del parametro SOURCE. Se tale parametro non può essere risolto in un nomefile o in un indirizzo web, l'astrattore, all'atto dell'inserimento, considererà come nomefile la stringa inalterata ma compilerà una stringa di warning che cercherà di spiegare i motivi della non risolvibilità. In caso contrario l'astrattore risolverà il nome del file e non genererà alcun warning.

In ogni caso l'astrattore controllerà se la pagina web (nel caso sia stata risolta) esista già tra le pagine HTML o tra quelle server. Se non esiste verrà creata una nuova pagina web e, solo nel caso si tratti di una pagina risolvibile, una pagina HTML o Server, a seconda dell'estensione. Verranno aggiunti anche i parametri eventuali Warning (per pagina web), nomefile e interna=falso per default (per pagina Server o HTML).

Solo ora sarà possibile inserire un'istanza alla tabella Frame con i parametri LINEA, NOME e l'identificativo della pagina web creata o individuata che rappresenta il source del frame.

Tag APERTURA BLOCCO JAVASCRIPT

< APERTURA BLOCCO JAVASCRIPT >

<LINEA=*numero di linea*>
</APERTURA BLOCCO JAVASCRIPT>

Questo tag mappa il tag <SCRIPT language=Javascript […] > che si può trovare all'interno di codice HTML in una pagina web.
L'astrattore elaborerà tale tag generando un nuovo record nella tabella blocco javascript, con linea pari al valore ottenuto e pagina web pari all'id della pagina web in analisi.

Tag CHIUSURA BLOCCO JAVASCRIPT
< CHIUSURA BLOCCO JAVASCRIPT >

Questo tag corrisponde al tag </SCRIPT> di HTML. Può però verificarsi una situazione ambigua: in un file HTML è incontrato un tag di un altro linguaggio di script, diverso da Javascript. Tale tag verrà giustamente ignorato dal parser. Più avanti però si incontrerà il tag </SCRIPT>. Si vorrebbe a questo punto che il parser ignori anche questo tag, ma perché ciò sia possibile, è necessario che il parser mantenga una variabile booleana che indichi se è in apertura un blocco non riconosciuto o di non interesse.
L'astrattore, viceversa, darà a questo tag un'interpretazione semplice, analoga a quella che dà al tag CHIUSURA.

Tag APERTURA BLOCCO VBSCRIPT
< APERTURA BLOCCO VBSCRIPT >
<LINEA=*numero di linea*>
</APERTURA BLOCCO VBSCRIPT>

Questo tag corrisponde all'espressione <% [language=VBScript] che si può ritrovare in una pagina asp. La sua semantica è praticamente analoga a quella del tag APERTURA BLOCCO JAVASCRIPT, e ciò riguarda anche la gestione delle eccezioni.

Tag CHIUSURA BLOCCO VBSCRIPT

< CHIUSURA BLOCCO VBSCRIPT >

Questo tag corrisponde all'espressione %> che si può incontrare in una pagina asp. La semantica è analoga a quella del tag CHIUSURA BLOCCO JAVASCRIPT.

Tag INCLUSIONE MODULO JAVASCRIPT

< INCLUSIONE MODULO JAVASCRIPT >

<NOMEFILE='*Nomefile del modulo*'>

< /INCLUSIONE MODULO JAVASCRIPT >

Questo tag viene generato da un caso particolare del tag HTML script:

< SCRIPT language=javascript source=' *nome del modulo javascript'* >

In questo caso il codice javascript si trova su di un altro file, il cui nome é contenuto nel tag. Il parser adotterà il solito comportamento nei confronti dei nomi di file.

Il comportamento dell'astrattore sarà il seguente: cercherà di risolvere il nome del file, generando eventualmente un warning, poi inserirà un'istanza nella tabella modulo javascript (a meno che non sia già presente) e un'altra nella tabella inclusionemodulojavascript. L'attributo interno del modulo javascript sarà per ora settato per default a falso.

Tag REDIRECT BLOCCO JAVASCRIPT

< REDIRECT BLOCCO JAVASCRIPT >

<LINEA=*numero di linea*>

<NOMEFILE='*Nomefile della destinazione*'>

< /REDIRECT BLOCCO JAVASCRIPT >

Questo tag viene generato dal comando location.href che si può incontrare nell'ambito di codice javascript. Il parser si limiterà a riportare il parametro nomefile così come leggibile dal codice.

L'astrattore cercherà di risolvere il nome del file, dopodiché aggiornerà o inserirà nella tabella pagina web la pagina destinazione (eventualmente anche nelle tabelle Pagina Server o Pagina HTML). Come sempre, nel caso in cui non sia possibile risolvere il nome del file, verrà aggiunto anche il campo warning con delle

indicazioni sulla motivazione. Infine aggiungerà un'istanza alla tabella RedirectBJ con gli identificativi della pagina o modulo in analisi e della pagina destinazione.

Tag REDIRECT MODULO JAVASCRIPT

< REDIRECT MODULO JAVASCRIPT >

<LINEA=*numero di linea*>

<NOMEFILE='*Nomefile della destinazione*'>

< /REDIRECT MODULO JAVASCRIPT >

Questo tag va a mappare ancora il comando location.href di Javascript, ma nelle occasioni in cui venga incontrato nell'ambito di un modulo Javascript. Il comportamento dell'astrattore sarà analogo al caso del tag REDIRECT BLOCCO JAVASCRIPT, laddove si consideri il modulo javascript invece del blocco javascript.

Tag REDIRECT BLOCCO VBSCRIPT

< REDIRECT BLOCCO VBSCRIPT >

<LINEA=*numero di linea*>

<NOMEFILE='*Nomefile della destinazione*'>

< /REDIRECT BLOCCO VBSCRIPT >

Questo tag si riferisce al metodo redirect(*nomefile*) che si può incontrare nell'ambito di codice VBScript. Le problematiche sono ancora quelle descritte per gli altri redirect, per cui anche il comportamento dell'astrattore è analogo.

Tag PAGINA CLIENT COSTRUITA

< PAGINA CLIENT COSTRUITA>

La sintassi di questo tag prevede un unico rigo d'informazione in quanto, in realtà, può esistere al più una pagina client costruita per ogni pagina asp (se per caso l'astrattore dovesse incontrare tale tag nell'ambito di una pagina HTML segnalerebbe un errore). La fine di tale pagina client costruita non può che coincidere con la chiusura della pagina asp che la genera, per cui sarebbe stato inutile un ulteriore tag di chiusura della pagina client costruita.

L'importanza di questo tag è quindi legata alla necessità di riconoscere se una certa pagina asp generi o meno una pagina client. Si sono riconosciuti due distinte condizioni sufficienti a poter rilevare quest'evento:

- quando, nell'ambito di una pagina asp si incontra il tag HTML <BODY>, che ci sta a simboleggiare la presenza di un corpo HTML nella pagina asp;
- quando, nell'ambito di un blocco VBScript, si incontra un metodo Response.Write(*stringa*) il quale indica che la stringa verrà scritta nella pagina generata.

Il comportamento conseguente dell'astrattore sarà quello di inserire un'istanza nella tabella pagina web ed una nella tabella pagina client costruita, settando ovviamente il valore di pagina server a quella attualmente in analisi.

Tag INTERFACCIA

< INTERFACCIA >

<LINEA=*numero di linea*>

<TIPO='*Tipo oggetto interfacciato*'>

</INTERFACCIA>

Questo tag viene inserito nel file intermedio dal parser qualora, nell'ambito di un blocco VBScript, si incontri un metodo del tipo Server.CreateObject(*interfaccia*). Il parser ricopierà la stringa interfaccia senza effettuare alcuna operazione.

L'astrattore viceversa tenterà di interpretare questa stringa e di ricondurla ad uno dei tipi predefiniti (ADODB.Connection, ADODB.Recordset, CDONTS.Newmail). Se ciò non è possibile, l'interfaccia verrà considerata di tipo generico, laddove generico può significare sia sconosciuto che non risolto.

L'astrattore aggiungerà quindi un'istanza alla tabella oggetto interfaccia, specificando il tipo trovato, poi un'istanza alla tabella interfaccia che collega le tabelle pagina server e oggetto interfaccia.

Tag APERTURA FORM

< APERTURA FORM>

<LINEA=*numero di linea*>

<METODO=GET | POST>

[<ACTION='*Nomefile della pagina a cui vengono inviati i dati*'>]

</ APERTURA FORM >

Questo tag proviene direttamente dal tag HTML la cui sintassi è:
< FORM [METHOD=GET|POST] [ACTION=' *'Nomefile della pagina a cui vengono inviati i dati*'] […]>
Come si può vedere entrambi i parametri sono facoltativi, ma, se l'assenza di METHOD fa intendere che deve essere preso il valore di default, l'assenza di ACTION, viceversa, ci informa che si tratta di un form di solo output. Come sempre il parser deve raccogliere tali dati, che possono anche essere in un altro ordine o misti ad altri parametri che non sono però di nostra pertinenza. Per il parametro action, il parser come sempre non cercherà di risolvere il nome del file.
L'astrattore dovrà prima di tutto cercare di risolvere il nome del file cui si riferisce il parametro action. A questo punto inserirà un nuovo record nella tabella pagina web (o aggiornerà un record esistente) e, eventualmente in pagina server o in pagina client (secondo la solita procedure per inserire file di destinazione). A questo punto si può inserire un'istanza alla tabella form e, se c'era un parametro action, alla tabella submit che punterà alla pagina web appena creata.

Tag CHIUSURA FORM
< CHIUSURA FORM>

Questo tag corrisponde esattamente al tag HTML </FORM> e serve solo come informazione all'astrattore di considerare esauriti i parametri della form aperta.

Tag PARAMETRO
< PARAMETRO>
    <TIPO=*tipo* >
    [<NOME=*'Nome assegnato al parametro'*>]
</PARAMETRO>

Questo tag corrisponde ai tag HTML <INPUT>, <SELECT>, <TEXTAREA>, che identificano un campo di input di una form. Se si incontra uno dei tag HTML generatori all'esterno della zona di influenza di una form, il tag non sarà preso in considerazione dal parser. Il parser non si occuperà nemmeno in questo caso del

riconoscimento del tipo, anche se stavolta i valori possibili sono finiti, in quanto potrebbe essere non risolvibile a causa di una parametrizzazione ottenuta con asp.
L'astrattore cercherà quindi di risolvere il tipo del parametro. I valori che si possono trovare sono HIDDEN, TEXT, PASSWORD, CHECKBOX, RADIO, SUBMIT, RESET, IMAGE, TEXTAREA, SELECT e GENERICO, laddove GENERICO può essere tanto un segnale che indica un errore di sintassi nell'HTML, tanto un nome non risolvibile.
Sul campo name non vengono fatte elaborazioni in quanto esso è solo un'etichetta e, a causa della possibile non risolvibilità, non si è tenuto conto di esso come possibile chiave primaria per la tabella parametri.
L'astrattore interagirà col database inserendo un'istanza alla tabella parametri, indicando anche l'identificativo del form attualmente in analisi.

Tag APPLET

< APPLET>
    <LINEA=*numero di linea*>
    <NOMEFILE='*Nome del file che contiene la classe*'>
</APPLET>

Corrisponde all'omonimo tag HTML. Il compito del parser è come al solito di registrare il numero di linea e la stringa che appare come nome del file.
L'astrattore viceversa cercherà di risolvere il nome del file. Se ciò non fosse possibile verrà generato un warning. Poi si inserirà nel database prima l'istanza nella tabella javaclass (a meno che non fosse presente già presente), poi nella tabella applet.

Tag IMMAGINE

< IMMAGINE>
    <LINEA=*numero di linea*>
    <NOMEFILE='*Nome del file sorgente*'>
</IMMAGINE>

Corrisponde al tag HTML <IMG >. La situazione è per il resto la stessa delle applet, per cui l'astrattore si comporterà in maniera perfettamente analoga, rispetto alle tabelle immagine e Ancora ad immagine.

Tag ANCORA

< ANCORA>

<LINEA=*numero di linea*>

<NOMEFILE='Nome del file della destinazione'>

</ANCORA>

Corrisponde al tag HTML <A HREF='*'Nome del file della destinazione*' […] >. Il parser come sempre non interpreterà l'attributo nomefile.

Il compito dell'astrattore sarà invece abbastanza complesso. Prima di tutto cercherà di risolvere l'indirizzo. Si possono verificare tutta una serie di casi:

- l'indirizzo comincia con una tra le parole chiave mailto, telnet, news. In questo caso l'astrattore aggiungerà un'istanza alla tabella ancora ad altro indirizzo;
- nell'ambito dell'indirizzo viene riconosciuto un nome di file la cui estensione è diversa da htm, html, asp. In questo caso si deve presumere si tratti di un ancora ad un oggetto scaricabile (il cui nome file è quello individuato). L'astrattore inserisce allora un'istanza nella tabella oggetto ed una nella tabella ancora ad oggetto;
- si riesce a riconoscere l'indirizzo di una pagina HTML o ASP: in tal caso l'astrattore crea un'istanza in pagina web, un'altra in pagina client oppure pagina server, infine una in ancora;
- non si riesce a risolvere il nome a causa della presenza di uno spezzone di asp (racchiuso tra <% e %>). In questo caso si crea una nuova pagina web, allegando però un messaggio di warning, poi un'istanza della tabella ancora che punta a questa pagina web;
- in tutti gli altri casi si presume di aver trovato l'indirizzo di un'applicazione web (come noto sono permesse parecchie sintassi possibili: ad esempio con o senza il prefisso http://, con o senza il prefisso www, etc.) e quindi si crea un'istanza nella tabella pagina web e una nella tabella ancora.

CAPITOLO 10 - IL TOOL ESTRATTORE

1 Introduzione

In questo capitolo si vogliono raccogliere tutti i requisiti descritti nell'analisi del capitolo 7, nella definizione del database del capitolo 8 e nella definizione della sintassi del file taggato del capitolo 9, per giungere alla progettazione del tool estrattore.

2 Analisi dei requisiti del tool estrattore

Molti dei paragrafi di quest'analisi dipendono direttamente dall'analisi dei requisiti del tool completo, che è stata definita nel capitolo settimo.
In particolare, il campo di applicazione è stato già specificato, mentre lo scopo di questo sistema è una parte dello scopo totale.
Il tool estrattore si propone come scopo quello di realizzare, in maniera automatica, la fase di ricerca delle informazioni statiche nel codice, così come indicata nella metodologia del capitolo 6.
Le informazioni ricavate dal codice devono essere sintetizzate in appositi file taggati che seguono la sintassi definita nel capitolo 9, in modo da poter essere coerentemente memorizzati nel database definito nel capitolo 8.
Tra gli scopi di questo tool non figura alcuna forma di rappresentazione grafica delle informazioni ricavate; tali astrazioni sono lasciate a tool successivi, i cui requisiti sono stati descritti nel capitolo 7, e che hanno trovato una loro implementazione nella tesi [1].

2.1 Descrizione generale

2.1.1 Funzionalità esterne del tool

Il tool assolve sostanzialmente ad una sola funzionalità: in ingresso ha una lista di tutti i file che si vogliono analizzare, in uscita restituisce:

- Una struttura di directory eguale a quella dell'applicazione web, dove ad ogni file analizzato corrisponde il file contenente le informazioni estratte. Questo file ha lo stesso nome di quello originario, con l'aggiunta di un'estensione .txt;
- Un file contenente la lista completa di tutti i file di cui si parla al punto precedente. Questo file viene denominato listaout.txt.

2.1.2 Caratteristiche dell'utente

Gli utenti sono i medesimi indicati in fase di analisi generale, nel capitolo 7.

2.1.3 Vincoli per l'utilizzo

L'unico vincolo significativo è dato dalla presenza di un file contenente la lista precisa di tutti i file dell'applicazione web da analizzare. Un modo di realizzare velocemente tale file è descritto nella guida per l'utente, nel caso esso operi in ambiente MS-DOS o Windows.

Inoltre devono essere disponibili, sulla macchina dove viene eseguito il tool, tutti i file sorgenti dell'applicazione web.

2.2 Specifica dei requisiti

I requisiti del sistema sono stati già ampiamente trattati nei capitoli precedenti. In particolare, è nota la sintassi che devono rispettare tutti i file risultanti dall'esecuzione del tool. Non ci sono altri requisiti particolari, riguardanti la sicurezza oppure la velocità.

3 Progetto dell'estrattore

Per la descrizione formale del progetto si utilizza il formalismo UML, in particolare class diagram e statechart diagram.

Essendo questo tool, essenzialmente, un automa riconoscitore, si è sviluppata un'unica classe, i cui metodi implementano dei sottoautomi che riconoscono particolari tipologie di tag o istruzioni del codice sorgente

3.1 Class diagram

Si riporta qui l'unico class diagram, nel quale figura l'unica classe prodotta, Estrattore, unitamente alla classe File, predefinita nel linguaggio.

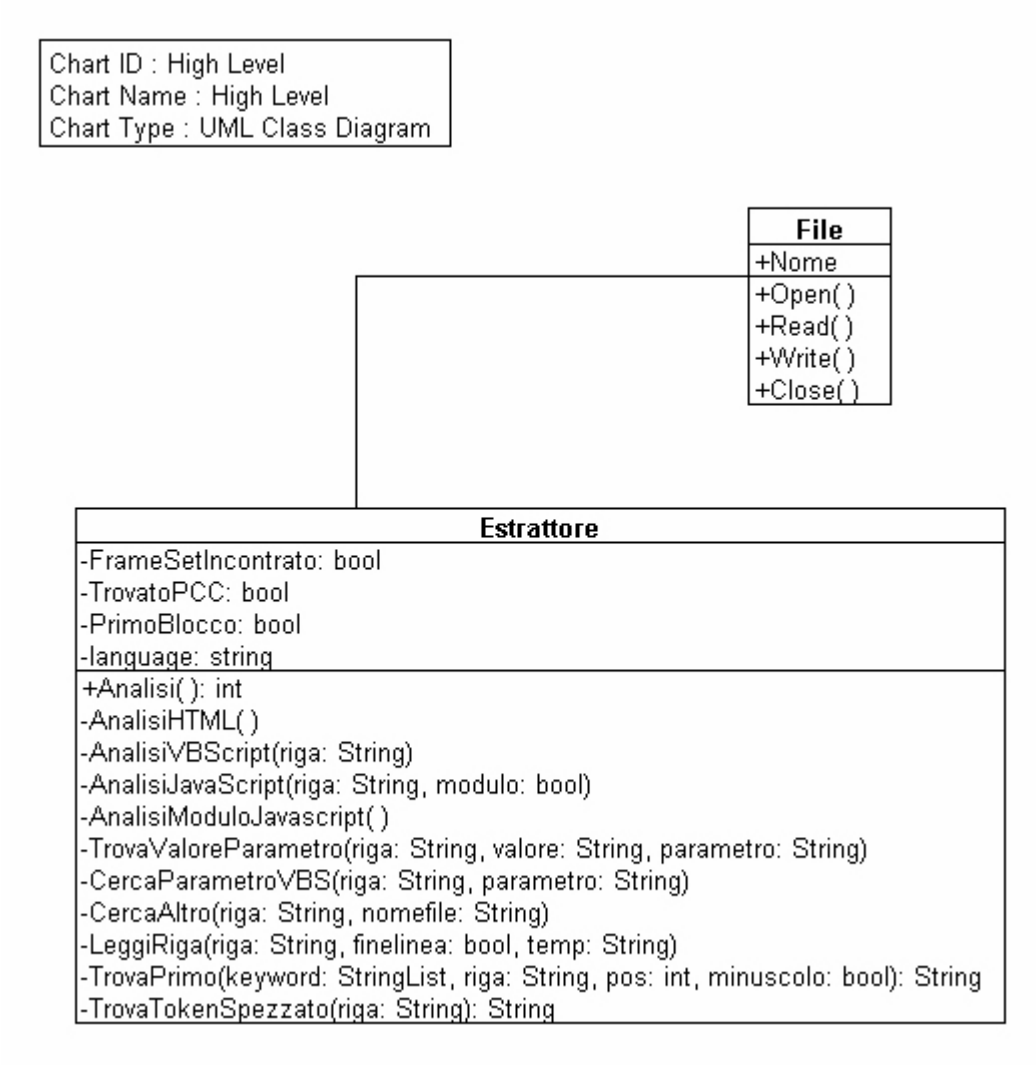


Figura 1 – Class diagram del tool estrattore

E’ necessario che venga istanziato sempre un unico oggetto della classe estrattore, in quanto il suo unico metodo pubblico, Analisi(), si occupa di tutte le elaborazioni di parsing sul fine in ingresso, nonché della produzione del file d’uscita.
L’unica classe esterna utilizzata è la classe file della quale vengono istanziati due oggetti: FileInput, che viene aperto in sola lettura, e FileOutput, che viene aperto in sola scrittura.

3.2 Progettazione delle classi

Gli strumenti che si utilizzeranno per rendere una chiara descrizione del tool e degli algoritmi seguiti in fase di progettazione e implementazione, saranno essenzialmente due: PDL, per la descrizione sommaria degli algoritmi, e Statechart diagrams, per la descrizione degli automi.

Classe estrattore

Metodo pubblico Analisi

Si tratta dell'unico metodo pubblico della classe, quindi dell'unico candidato ad essere richiamato da classi esterne. Ecco una breve descrizione PDL dell'algoritmo seguito:

```
Apri file di input
Ricava percorso e estensione
Apri file di output
Scrivi sul file di output il tag di apertura
IF (estensione=html)or(estensione=asp)
     AnalisiHTML
ELSE IF (estensione=js)
     AnalisiModuloJavascript
END IF
Scrivi sul file di output il tag di chiusura
```

Metodo privato AnalisiHTML

Questo metodo consiste nell'implementazione di un automa che scandisce il file di input in cerca di tag HTML, cedendo quindi il controllo ad altri metodi qualora si incontri un tag HTML rilevante. Si esce da questo metodo solo quando si incontra la fine del file.

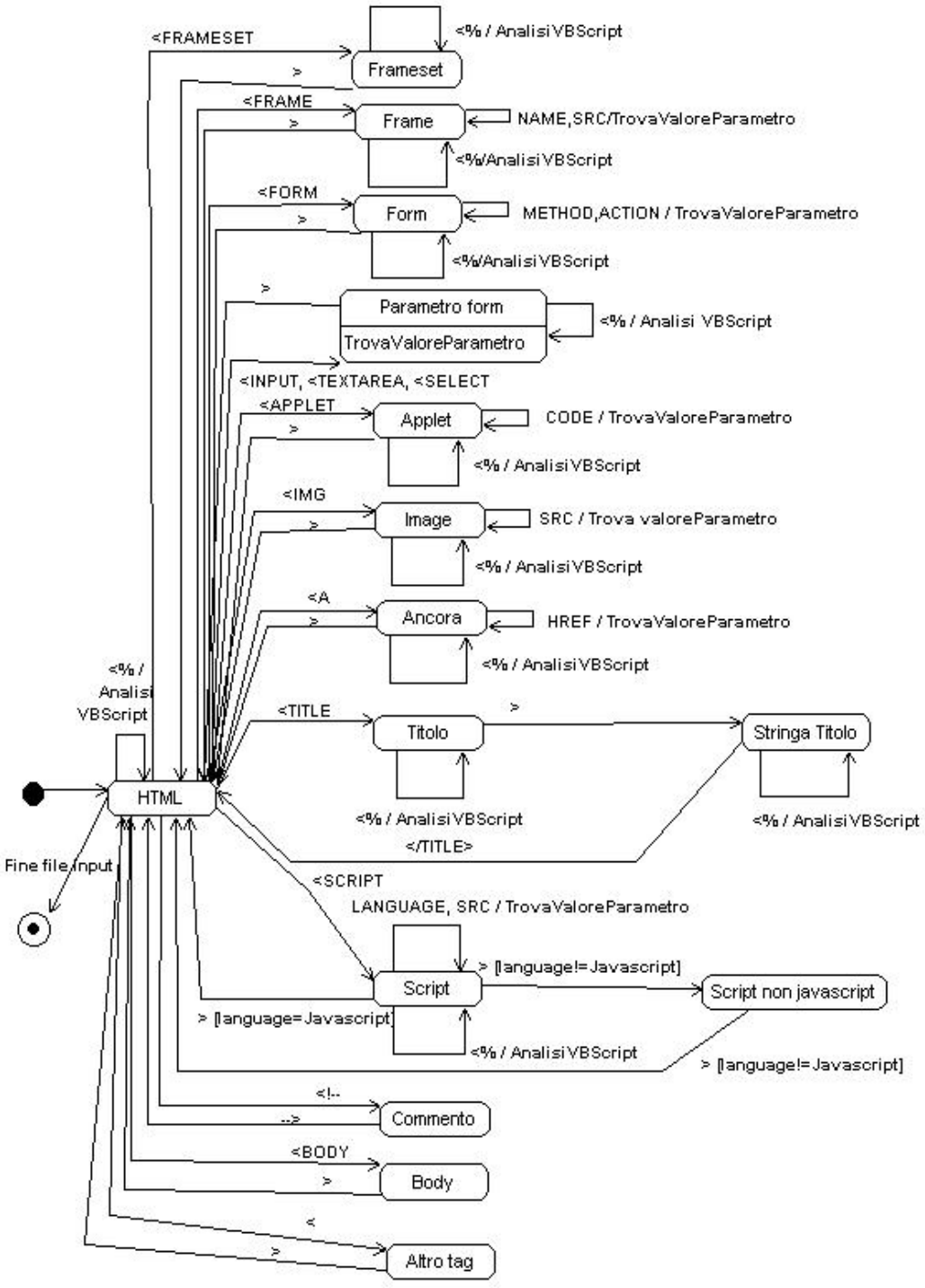


Figura 2 - Automa riconoscitore di tag HTML implementato nel metodo AnalisiHTML

Metodo privato AnalisiVBScript (String riga)

Anche questo metodo è descrivibile come un automa che scandisce il file di input a partire dalla riga fornita in ingresso, in cerca di comandi VBScript rilevanti.

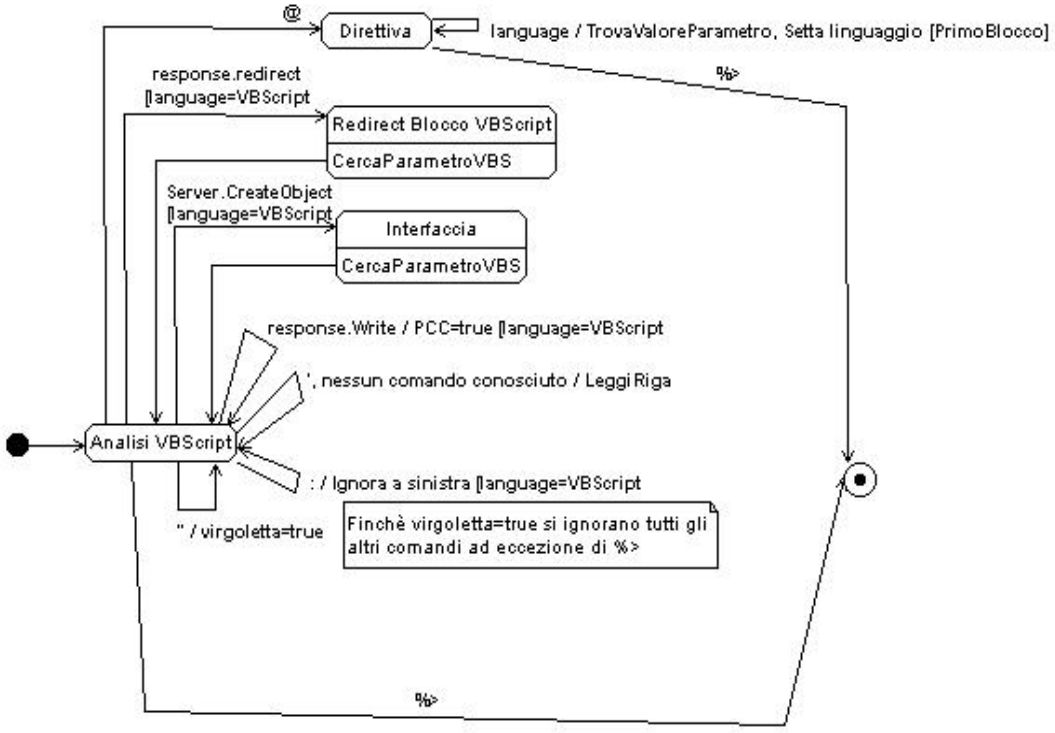


Figura 3 - Statechart raffigurante l'automa che riconosce comandi VBScript, implementato nel metodo AnalisiVBScript

Quest'automa viene seguito solo se il flag che indica il linguaggio è settato al valore VBScript (che è anche il valore di default), altrimenti si ignora tutto il blocco di script. Da notare che il linguaggio di script si può settare solo nel primo blocco script di un file asp.

Metodo privato AnalisiJavascript (String riga)

Anche questo metodo verrà descritto con uno statechart diagram.

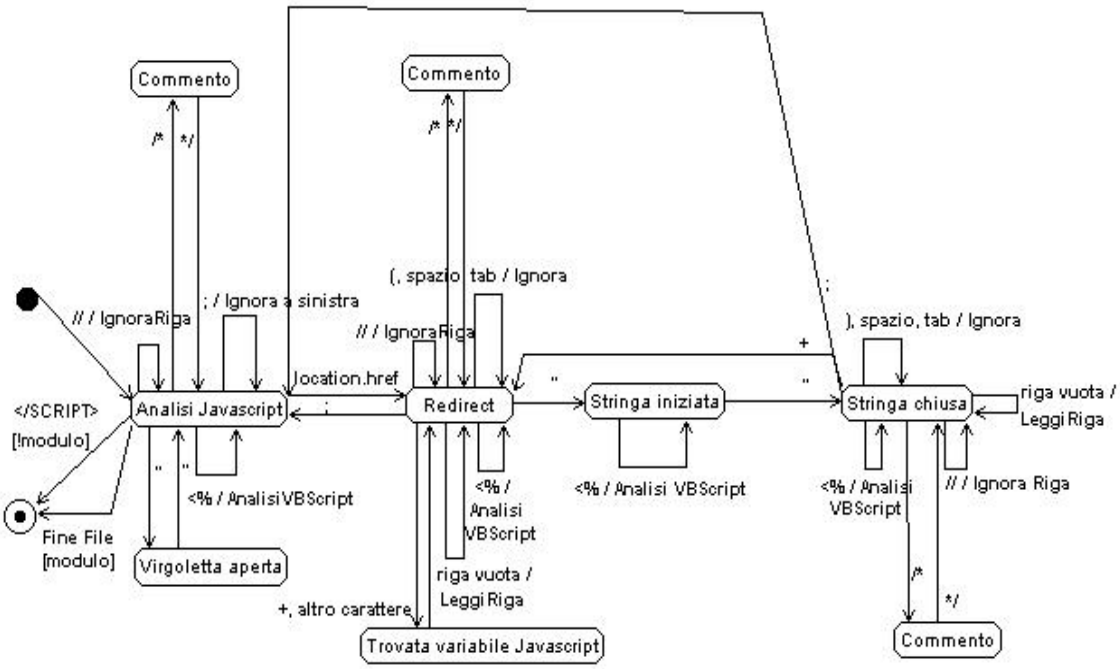


Figura 4 - Statechart raffigurante l'automa riconoscitore di parole chiave Javascript, implementato nel metodo AnalisiJavscript

Come si può vedere, l'unico comando Javascript che viene riconosciuto è location.href, per il quale deve essere trovato l'argomento, riconoscendo in esso le parti costanti e quelle variabili.

Si noti che sono possibili due diversi eventi che causano l'uscita:

- se il metodo è stato richiamato da Analisi HTML si esce quando si incontra il tag </SCRIPT>;
- se il metodo è stato richiamato da Analisi Modulo Javascript (ovvero stiamo scandendo un file con estensione .js) si esce solo al termine del file.

Metodo privato AnalisiModuloJavascript

Questo metodo molto semplice viene richiamato quando c'è da esaminare un modulo scritto in Javascript e si limita a leggerne la prima riga e richiamare AnalisiJavascript, notificandogli che si tratta dell'analisi di un modulo Javascript, anziché di un blocco.

Metodo privato TrovaValoreParametro (riga : String, valore:String, input parametro:String)

Questo metodo intercetta il valore di un parametro di un tag HTML che si trova a partire dalla riga in input (in HTML carriage return è completamente assimilato a spazio e tab come separatore), individuandone le parti costanti e le parti variabili.

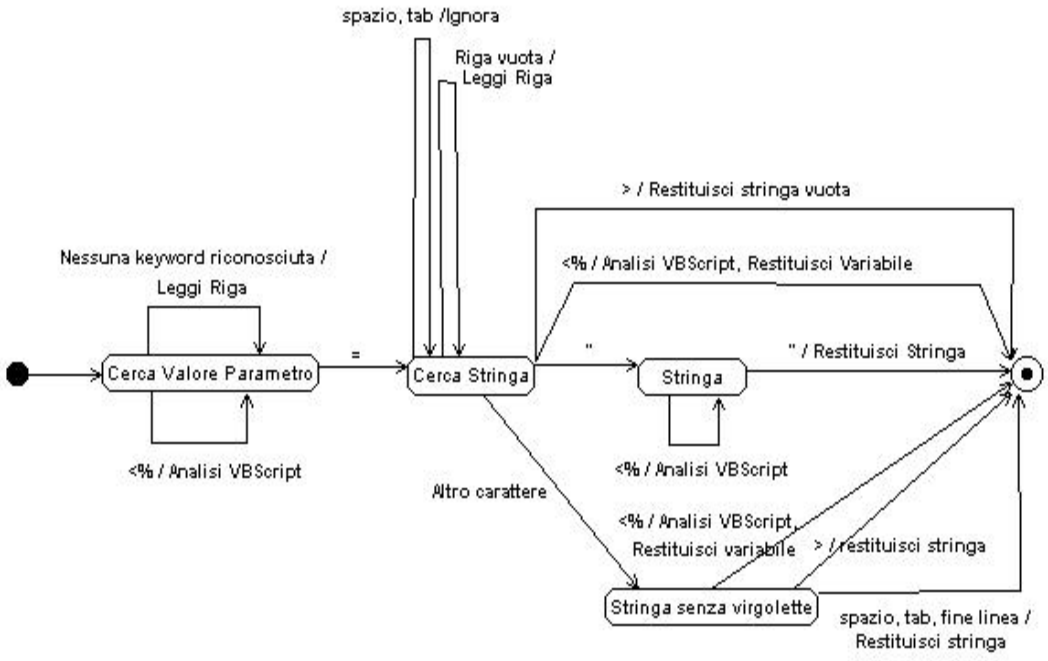


Figura 5 - Statechart raffigurante l'automa che riconosce un parametro di un tag HTML, implemento net metodo TrovaValoreParametro

Metodo privato CercaParametroVBS (riga : String, output parametro: String)

Questo metodo intercetta il valore di un parametro di un comando VBScript, distinguendo come sempre le parti costanti da quelle variabili. Si è tenuta in considerazione anche la particolare sintassi di VBScript, che prevede di poter anche dividere una stringa in più righe, se però la riga si interrompe con un underscore. Questo metodo si occupa in particolare di riconoscere stringhe, demandando il problema di riconoscere variabili VBScript al metodo CercaAltro.
L'automa realizzato è il seguente:

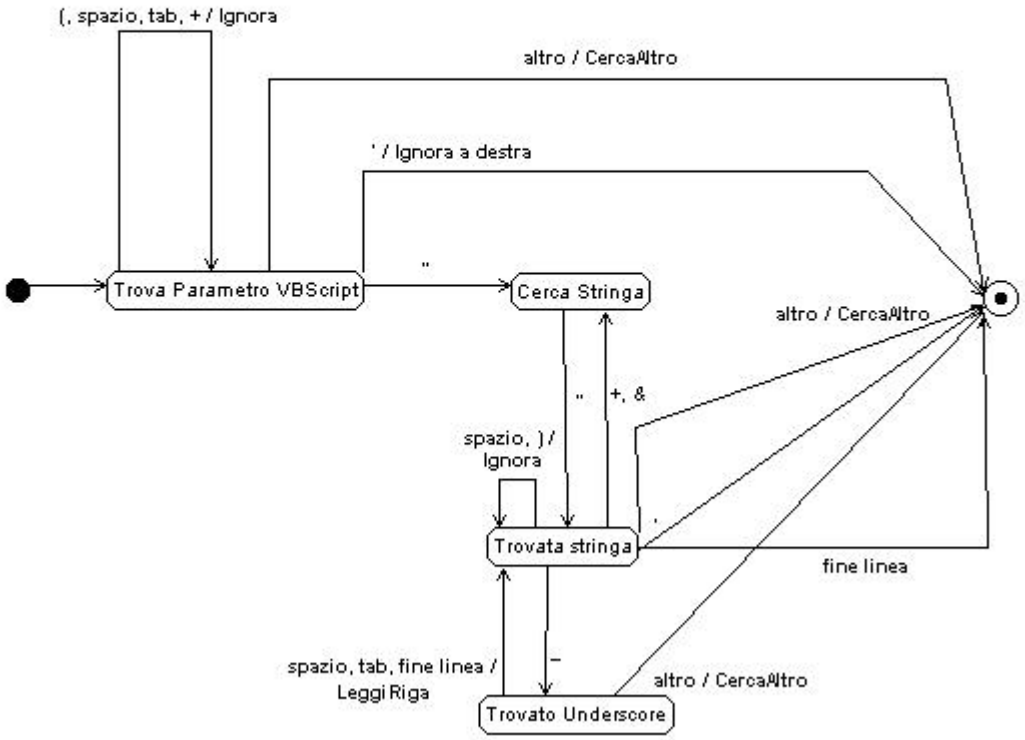


Figura 6 - Statechart raffigurante l'automa che riconosce un parametro di un'istruzione VBScript

Metodo privato CercaAltro (riga: String, nomefile : String)

Questo metodo è d'ausilio a CercaParametroVBS ed ha il compito di completare il nomefile presente sulla riga ed elaborato, nella sua parte costante, da CercaParametroVBS. Si vogliono qui riconoscere eventuali nomi di variabili VBScript, in modo da poter fornire all'utente qualche informazione anche sul valore dinamico di questo parametro.
Ecco l'automa:

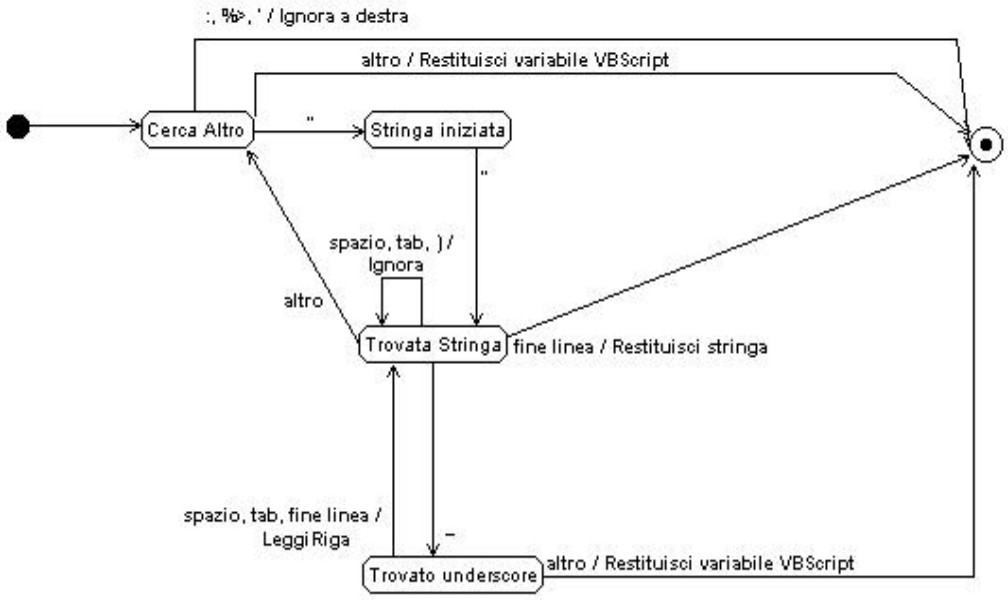


Figura 7 - Statechart raffigurante un automa che coadiuva l'automa che riconosce i parametri di un istruzione VBScript, implementato nel metodo CercaAltro

Metodo privato LeggiRiga (input temp:String , finelinea: bool, output riga:String)

Questo metodo legge una riga dal file in input ed aggiorna anche il contatore delle linee lette. Per comprendere il significato degli altri parametri finelinea e temp, vedi le note sull'implementazione.

Metodo privato TrovaPrimo (input bool minuscolo, StringList keyword, String riga , output int posizione): String

Questo metodo è stato implicitamente utilizzato da tutti gli automi che scandiscono il file di input. Esso cerca quale sia, tra le keywords inserite nella StringList, la prima a presentarsi, cancellando dalla riga tutto ciò che viene prima della keyword. E'possibile che alcune keywords siano sottinsiemi di altre (ad esempio </ e </SCRIPT>). In tal caso il metodo restituisce la più lunga tra queste. Il metodo restituisce anche la posizione nella riga originaria della keyword trovata e la keyword trovata. Nell'utilizzo di questo metodo e dei valori che restituisce, è molto importante tener conto degli effetti collaterali distruttivi che esso può introdurre sulla riga.
Se non viene trovata alcuna Stringa il metodo lascia inalterata riga, setta posizione a –1 e restituisce una stringa vuota.

Metodo privato TrovaTokenSpezzato (riga:String) : String

Questo semplice metodo trova quello che può essere un token che sia stato inavvertitamente spezzato nella suddivisione in linee che si è fatta del file in input (vedi note sull'implementazione). Restituisce tutto ciò che si trova a destra dell'ultimo carattere separatore (spazio oppure tab), oppure tutta la riga se non ci sono separatori.

4 Note sull'implementazione

In questo paragrafo si vogliono includere tutte quelle altre informazioni che, pur non facendo parte della progettazione concettuale del tool, sono necessarie per una sua corretta comprensione e manutenzione.

4.1 Lettura del file in input

La procedura più semplice e naturale per leggere il file di input è sicuramente quella di leggerlo per righe (laddove l'elemento separatore delle righe è il carriage return). D'altronde si tratta di un accorgimento importante in quanto, tra i linguaggi da esaminare, VBScript adotta, tra i separatori delle istruzioni, appunto il carriage return. Purtroppo, però, la sintassi HTML non vieta l'esistenza di linee molto lunghe, con più di 256 caratteri, per cui non è sempre possibile assegnare ad una stringa (secondo l'implementazione del tipo CString) il valore di una riga. Si è dovuto escogitare quindi un sistema alternativo, che spezza le linee o con i carriage return, oppure al 128-esimo carattere.

Rimane importante poter distinguere tra le righe che sono terminate anche nel file e quelle che invece sono state spezzate: è a questo scopo che è adibito il booleano finelinea cui si è già accennato. Bisogna quindi intendere l'operazione LeggiRiga che si può leggere in molti Statechart diagram, come un'operazione che viene effettuata solo al termine effettivo di ogni riga, altrimenti ci si limita a leggere una sottoriga e continuare.

A questo punto si presenta un ulteriore problema, che è quello dei token spezzati, ovvero delle parole chiave che vengono a trovarsi divise, parte in una sottoriga e parte in una successiva. E' appunto il metodo TrovaTokenSpezzato che risolve il problema, riconoscendo come token spezzato tutto ciò che viene dopo l'ultimo carattere separatore premettendolo alla successiva sottoriga che viene letta. E' per evitare che la lunghezza della stringa ottenuta dall'unione del token spezzato e della sottoriga sia superiore a 256 caratteri che si è scelto di porre la lunghezza delle sottorighe a 128 caratteri.

4.2 Attributi privati della classe Estrattore

La classe Estrattore necessita anche di un certo numero di attributi che hanno bisogno di essere visibili da molti metodi, perché fungono da variabili di stato oppure da risorsa comune.

Variabili di stato:

- bool TrovatoPCC: questo flag ci indica se si è già scritto nel file di output il tag <PAGINA CLIENT COSTRUITA>. Ovviamente ha rilevanza solo se si sta analizzando una pagina asp. Viene settato al valore vero dalla prima istruzione che implica una visualizzazione.
- Bool FramesetIncontrato: questo flag serve a rispettare una convenzione che era stata presa in precedenza, ovvero quella di considerare, in una pagina che indirizza verso i frame, solo la prima occorrenza del tag HTML <FRAMESET>, e di ignorare le ulteriori occorrenze che si nidificano. Il tag è quindi settato vero al primo frameset, ed indica di non generare output quando si incontrano i successivi tag frameset.
- Bool PrimoBlocco: questo flag ci indica se siamo in attesa di incontrare il primo Blocco di script server o ne abbiamo già incontrato uno (l'importanza deriva dalla possibilità, nel primo blocco di un documento, di settare il linguaggio di script).
- String language: indica il linguaggio di script server utilizzato nella pagina. Il valore predefinito è VBSCRIPT, ma è possibile settarlo nel primo blocco. In effetti in una pagina possono essere presenti blocchi di differenti linguaggi client, ma è possibile definire una sola volta il linguaggio server.

4.3 Variabili di utilizzo comune tra i metodi

- Int linea: contiene il numero di linea del file di input attualmente in analisi. E' utilizzato da tutti i metodi scanditori per scrivere la riga in cui si è incontrato il tag.
- Bool finelinea: questo flag è legato alla suddivisione in sottorighe delle righe logiche, descritta nel paragrafo precedente. Assume valore vero per le sottorighe che contengono anche la conclusione di una riga.

Gli altri attributi che compaiono risultano dalle assegnazioni dei parametri della chiamata nomefile e path che ci indicano appunto il nomefile e il percorso del file che si sta analizzando.
Ci sono infine i puntatori agli oggetti di tipo File: FileInput, FileOutput, FileLista.

5 Classe File

L'unica altra classe che è stata citata nel class diagram di figura 1 è la classe file, che corrisponde alla classe CStdIOFile predefinita nel linguaggio Visual C++. Trattandosi di una classe predefinita, non verranno trattati in dettaglio i metodi e gli attributi, rimandando per questo alla guida in linea MSDN, citata in bibliografia.
I metodi principali che sono stati utilizzati sono Open, Close, ReadString e WriteString, che permettono di aprire e chiudere il file, leggere e scrivere stringhe su di esso.
L'unico attributo significativo utilizzato è il nome del file.

6 Guida per l'utente

In questo paragrafo si vogliono includere tutta una serie di informazioni pratiche sull'utilizzo del tool.

Il tool è stato realizzato utilizzando l'ambiente di programmazione Visual C++ della Microsoft (versione 6.0). I test su questo programma sono state fatte sempre su di un sistema operativo Windows 98.

Preparazione all'esecuzione

Sono necessarie alcune operazioni preliminari all'esecuzione del tool. Si suppone qui che l'applicazione web da esaminare sia completamente contenuto in una cartella (che qui denomineremo *sito*, e nelle sue sotto cartelle, e che questa cartella sia disponibile sulla stessa macchina sulla quale opera il tool.

La prima operazione da effettuare consiste nel creare una lista dei file da analizzare. Per ogni file deve essere specificato il nome preceduto dal percorso completo, relativamente alla macchina sulla quale si trova.

Se si opera in ambiente MS-DOS o Windows e l'applicazione web è contenuta effettivamente in una cartella sito, si può eseguire, dal prompt di MS-DOS, la seguente linea di comando:

```
dir sito\*.* /b /s /a:-d >listasito.txt
```

dove *sito* è la cartella contenente l'applicazione web da analizzare e *listasito.txt* è il file su cui verranno salvate le informazioni.

E' consigliabile che il file *listasito.txt* non sia salvato all'interno dell'applicazione stessa, in modo da non alterarne la struttura.

Esecuzione

Il file eseguibile è parser.exe. Esso apre un menu con due scelte:

- Scegli sito, che permette di selezionare il file contenente la lista dei file dell'applicazione web, generato secondo la procedura indicata nel paragrafo precedente;

- Leggi file, che avvia l'estrazione delle informazioni. Quest'ultima operazione fa uso di una progress bar, la quale fornisce all'utente l'indicazione sull'effettivo avanzamento dell'analisi.

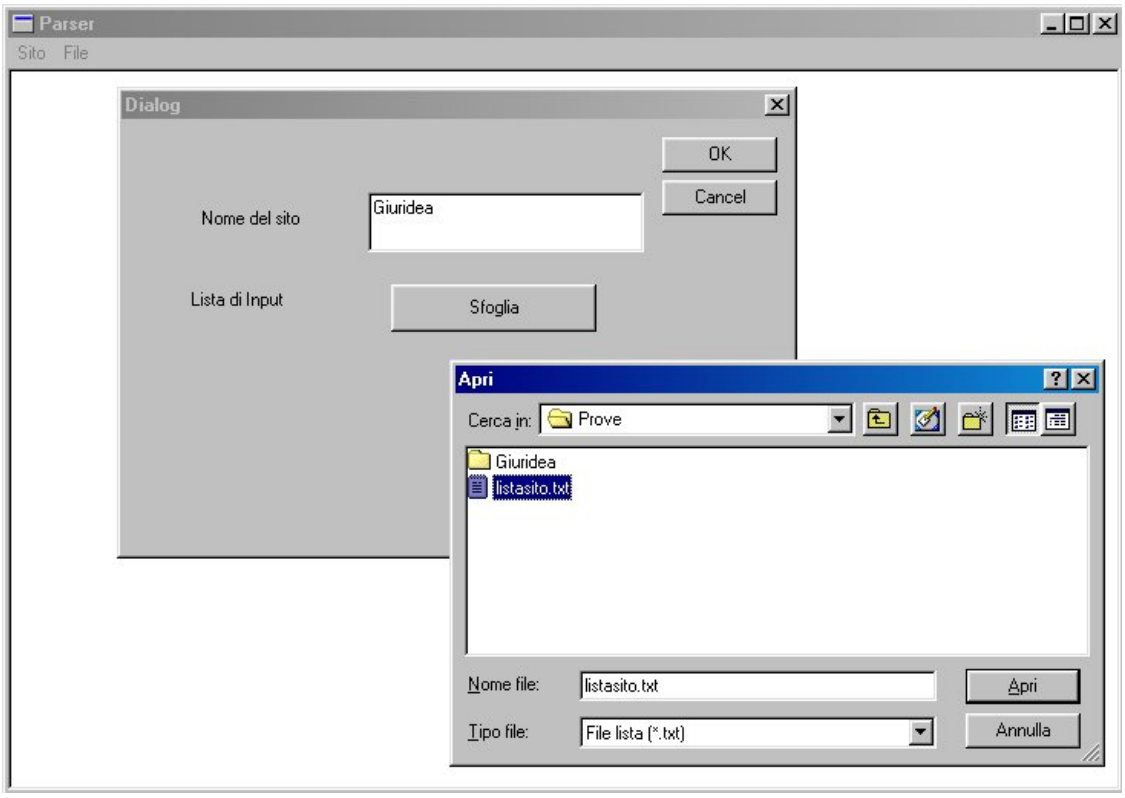


Figura 9 – Esecuzione del tool estrattore: scelta del sito web da analizzare

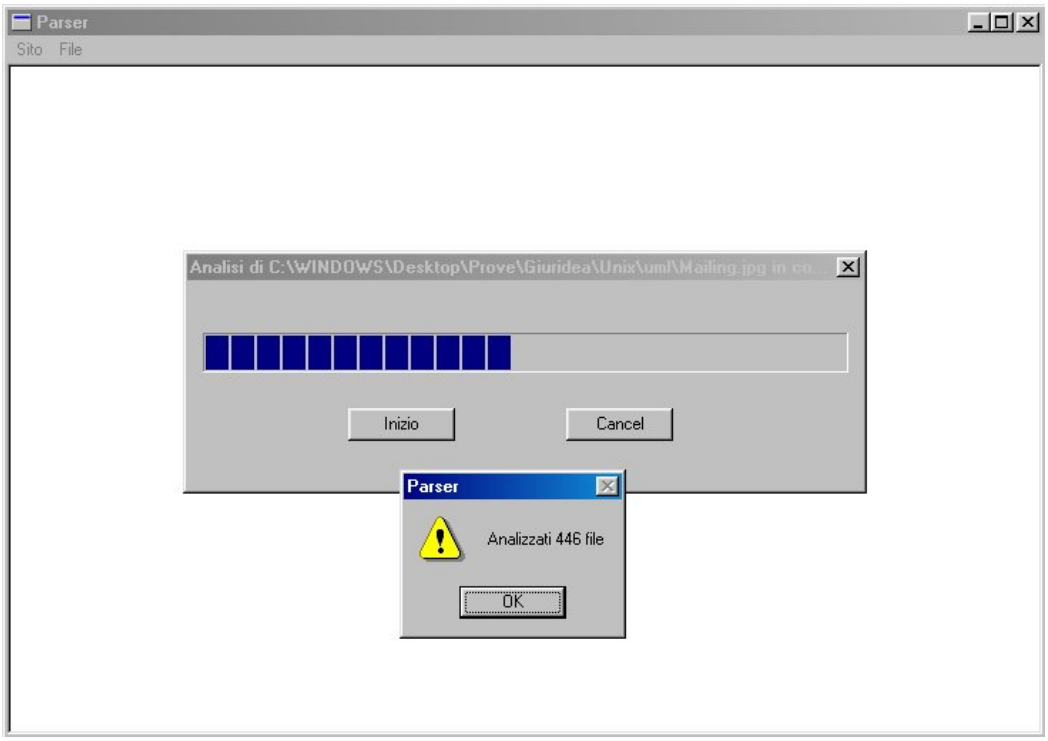


Figura 10 – Esecuzione del tool estrattore: estrazione delle informazioni

Il risultato dell'esecuzione consiste nel file *listasito.txt* e nella cartella *Output*, il cui contenuto è stato già descritto in precednza.

Figura 11 – Il risultato di una prova effettuata con il tool di estrazione

CAPITOLO 11 - CASI DI STUDIO

1 Introduzione

Questo capitolo è dedicato alla prova sul campo della metodologia di reverse engineering descritta nel capitolo 6.

Si sono scelti due casi di studio di dimensioni maggiori rispetto all'esempio del capitolo 5, in modo da avere conferme sull'effettiva utilità della metodologia ed apporvi eventualmente migliorie.

L'analisi si avvarrà dello strumento automatico di estrazione sviluppato. L'utilizzo di quest'ultimo porta ad una notevole riduzione del tempo necessario allo studio, ma anche ad alcune limitazioni dettate dal suo campo d'applicazione.

L'analisi delle applicazioni web seguirà il seguente schema, che ricalca quello teorizzato nel capitolo 6:

- Analisi preliminari sull'applicazione web: ricerca della pagina iniziale, analisi della scomposizione in directory, ipotesi euristiche sull'ordine di analisi dei file;
- Analisi statica dei file, possibilmente in un ordine che ricalchi la navigazione tra essi, e quindi le sottoparti dell'applicazione candidate ad essere funzionalità del sistema;
- Tracciamento di diagrammi. I diagrammi che vengono disegnati in questa fase sono il diagramma dei collegamenti, particolarmente utile per descrivere le parti statiche di un'applicazione web, e i class diagram UML, che invece si preferiscono per le componenti attive dell'applicazione.

2 Caso di studio: Giuridea

2.1 Analisi preliminari

Il primo caso di studio riguarda un'applicazione web realizzata dalla società Nablacom, il quale si propone come "Laboratorio Giuridico". Di quest'applicazione sono stati gentilmente forniti dallo sviluppatore i sorgenti. Inoltre, trattandosi di un'applicazione attualmente funzionante, è disponibile anche la sua versione on-line, all'indirizzo http://www.giuridea.it .

Come previsto dalla metodologia descritta nel capitolo 6, il primo passo consiste in un'analisi esterna del codice sorgente. L'applicazione web è scomposta in due parti, le quali sono in esecuzione su due differenti web server: La cartella NT utilizza il web server del sistema operativo Windows NT, mentre la cartella Unix usa il web server del sistema operativo Unix. Quest'informazione proviene direttamente da un colloquio con lo sviluppatore, del quale si è saputo anche che le parti su Unix comprendono pagine statiche oppure con interazioni limitate al lato client (in linguaggio Javascript), mentre le parti su Windows NT realizzano le funzionalità dinamiche, utilizzando la tecnologia ASP.

Tenendo presente questa separazione di contenuti, si inizia l'analisi dalla home page predefnita, index.htm.

2.2 Analisi statica

File index.htm

Visitando tale pagina, si vede come essa funge da punto di partenza per le maggiori funzionalità dell'applicazione. Una strategia promettente consiste nell'esaminare i collegamenti partenti da questa pagina e considerare ognuno di essi come punto di partenza di una funzionalità dell'applicazione.

Si vuole effettuare un'analisi statica della pagina e si utilizza, a tale scopo, il tool di estrazione realizzato. Viene qui riportato, solo per questa volta, il file risultante dall'analisi:

```
<APERTURA>
      <NOMEFILE="\index.htm">
</APERTURA>
```

```
<TITOLO>
      <TITOLO="Giuridea - Forum e Laboratorio Giuridico">
</TITOLO>
<APERTURA BLOCCO JAVASCRIPT>
      <LINEA=22>
</APERTURA BLOCCO JAVASCRIPT>
<CHIUSURA BLOCCO JAVASCRIPT>
<IMMAGINE>
      <LINEA=50>
      <NOMEFILE="images/title.jpg">
</IMMAGINE>
<ANCORA>
      <LINEA=51>
      <NOMEFILE="Archivio/Archivio.htm">
</ANCORA>
<IMMAGINE>
      <LINEA=51>
      <NOMEFILE="images/ArchivioHomeOff.gif">
</IMMAGINE>
<ANCORA>
      <LINEA=53>
      <NOMEFILE="Caso/Caso.htm">
</ANCORA>
<IMMAGINE>
      <LINEA=53>
      <NOMEFILE="images/CasoHomeOff.gif">
</IMMAGINE>
<ANCORA>
      <LINEA=55>
      <NOMEFILE="ChiSiamo/ChiSiamo.htm">
</ANCORA>
<IMMAGINE>
      <LINEA=55>
      <NOMEFILE="images/ChiSiamoHomeOff.gif">
</IMMAGINE>
<ANCORA>
      <LINEA=56>
      <NOMEFILE="Forum/Forum.htm">
</ANCORA>
<IMMAGINE>
      <LINEA=56>
      <NOMEFILE="images/ForumHomeOff.gif">
```

```
</IMMAGINE>
<ANCORA>
      <LINEA=57>
      <NOMEFILE="Novita/Novita.htm">
</ANCORA>
<IMMAGINE>
      <LINEA=57>
      <NOMEFILE="images/NovitaHomeOff.gif">
</IMMAGINE>
<ANCORA>
      <LINEA=58>
      <NOMEFILE="Specialisti/Specialisti.htm">
</ANCORA>
<IMMAGINE>
      <LINEA=58>
      <NOMEFILE="images/SpecialistiHomeOff.gif">
</IMMAGINE>
<ANCORA>
      <LINEA=59>
      <NOMEFILE="Cerca/Cerca.htm">
</ANCORA>
<IMMAGINE>
      <LINEA=59>
      <NOMEFILE="images/CercaHomeOff.gif">
</IMMAGINE>
<IMMAGINE>
      <LINEA=60>
      <NOMEFILE="http://www.giuridea.it/cgi/Count.cgi?dd=E|ft=0|tr=T
|trgb=ffffff|df=giuridea.it.00.dat">
</IMMAGINE>
<ANCORA>
      <LINEA=65>
      <NOMEFILE="mailto:webmaster@nablacom.it">
</ANCORA>
<ANCORA>
      <LINEA=66>
      <NOMEFILE="http://www.nablacom.it">
</ANCORA>
<IMMAGINE>
      <LINEA=66>
      <NOMEFILE="images/NABLACOM.jpg">
</IMMAGINE>
```

```
<ANCORA>
      <LINEA=67>
      <NOMEFILE="http://www.giuridea.it/">
</ANCORA>
<IMMAGINE>
      <LINEA=71>
      <NOMEFILE="images/NUOVO2.jpg">
</IMMAGINE>
<IMMAGINE>
      <LINEA=86>
      <NOMEFILE="images/NUOVO2.jpg">
</IMMAGINE>
<CHIUSURA>
```

Delle informazioni fornite dall'analizzatore, è opportuno trascurare quelle meno rilevanti, come le immagini e gli oggetti scaricabili inclusi in una pagina, al fine di ottenere il diagramma dei collegamenti descritto nel capitolo 4.

Le informazioni restanti sono state poi utilizzate come input per un programma visualizzatore di grafi, VCG. Il risultato è il seguente:

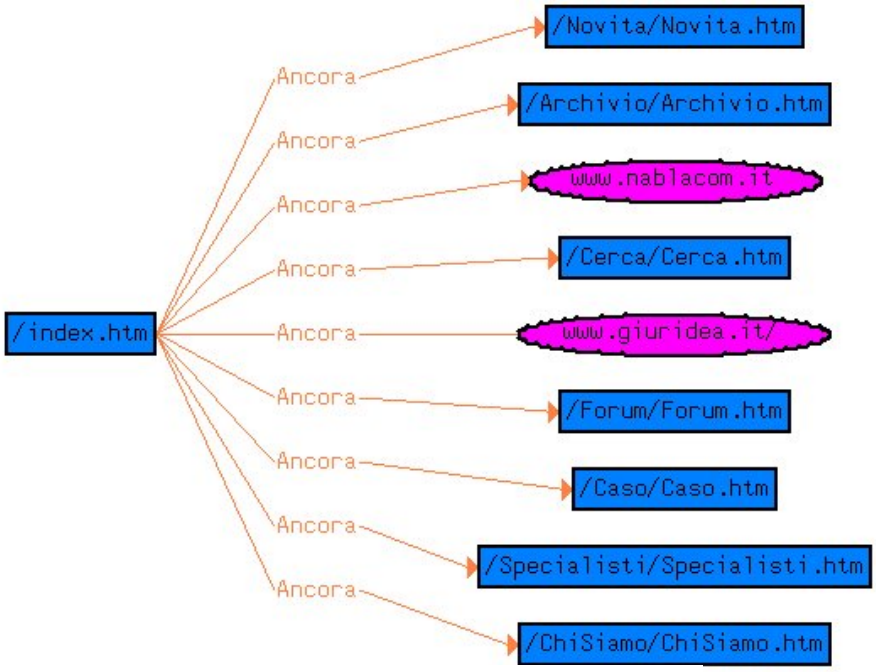


Figura 1 – Pagine collegate a index.htm

Si è convenuto di utilizzare rettangoli blu per le pagine HTML, ovali rosa per le pagine esterne all'applicazione web che si sta studiando.

Da questo primo diagramma si può notare come ognuna delle pagine linkate da index.htm si trova in una diversa sottocartella dell'applicazione web. L'ipotesi più naturale che si fa a questo punto, è che ognuna di queste pagine sia l'inizio di un percorso di navigazione compreso all'interno della directory nella quale si trova questa pagina. L'analisi si sposterà allo studio ordinato di queste subdirectory, a partire dalle pagine viste.

Cartella Novita

Il percorso di navigazione ipotizzato comincia con novita.htm. Si analizza tale file con l'estrattore automatico, poi, procedendo come nel caso precedente, si ricava il seguente grafo:

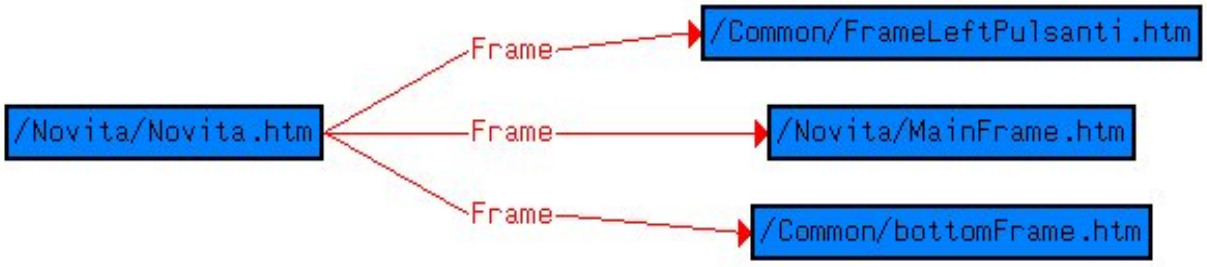


Figura 2 – Pagine collegate a Novita.htm

Si vede subito come la pagina novita.htm sia in realtà la sede di informazioni riguardanti la suddivisione in frame. Vedendo anche la versione on-line del sito, si può vedere come la pagina sia suddivisa in tre zone: una barra in basso (bottonFrame.htm), una barra a sinistra (FrameLeftPulsanti.htm) e una zona centrale.

Le barre si trovano in una directory Common: questa circostanza può essere indizio del fatto che esse siano comuni alle varie parti dell'applicazione, per cui il frame significativo di questo percorso è quello centrale. L'analisi si sposta quindi a MainFrame.htm.

File Mainframe.htm

C'è un'ulteriore suddivisione in frame. Si può vedere come il frame superiore (Title.htm) contenga solo un'immagine fissa, per cui il contenuto della pagina è in Text.htm

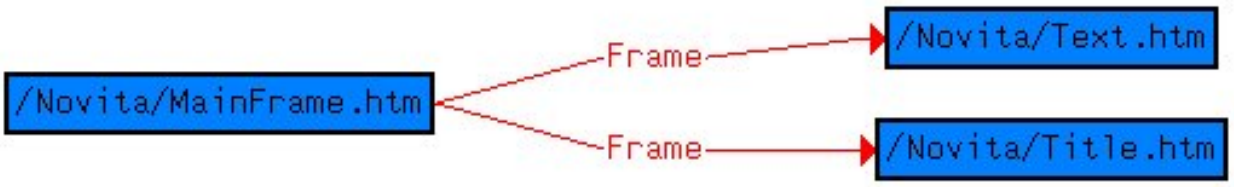


Figura 3 – Pagine collegate a MainFrame.htm

File Text.htm

Questa pagina contiene, oltre al testo, solo delle ancore verso altre pagine della stessa cartella. Lo studio deve ora proseguire verso queste pagine. L'analizzatore però trova che nessuna di queste pagine ha dei collegamenti verso altre pagine dell' applicazione, per cui si può considerare conclusa l'analisi di questa zona.

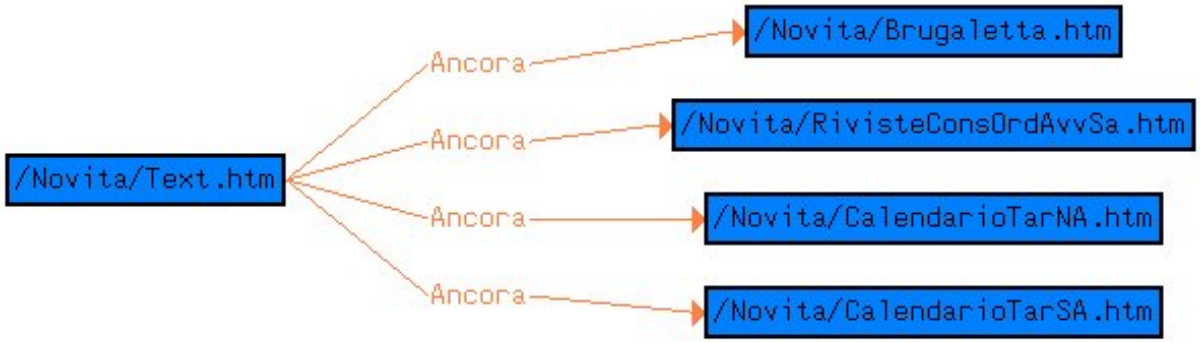


Figura 4 – Pagine collegate a Text.htm

Combinando i grafi precedenti, si può giungere ad un diagramma dei collegamenti complessivo:

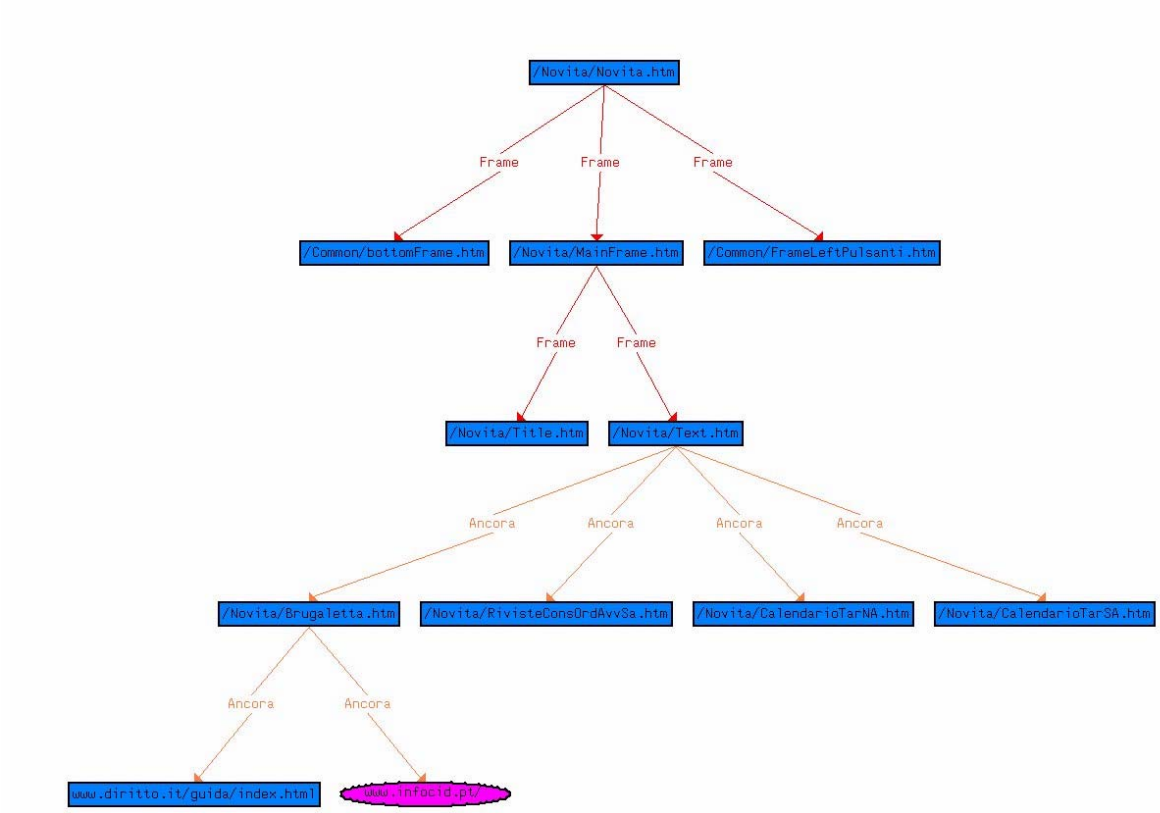


Figura 5 – Diagramma dei collegamenti a partire da Novita.htm

Concludendo lo studio di questa parte dell'applicazione si può vedere come, a parte i frame comuni, essa fornisca un percorso ipertestuale autosufficiente e passivo, seriamente candidato ad essere un caso d'uso del sistema.

Cartella Archivio

File Archivio.htm

Analizzando questo file si ottiene il seguente grafo:

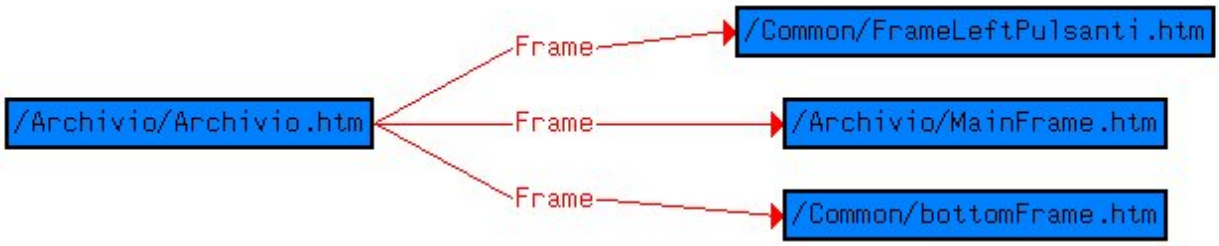


Figura 6 – Pagine collegate a Archivio.htm

Si può notare subito come la struttura sia la stessa del caso precedente, per cui si passa subito a MainFrame.htm, che però ha la stessa struttura del suo omonimo nella cartella Novita, per cui lo studio passa a text.htm. Questa pagina contiene solo dei collegamenti a degli oggetti scaricabili, per cui il percorso di analisi è già concluso e si può tracciare il grafo complessivo.

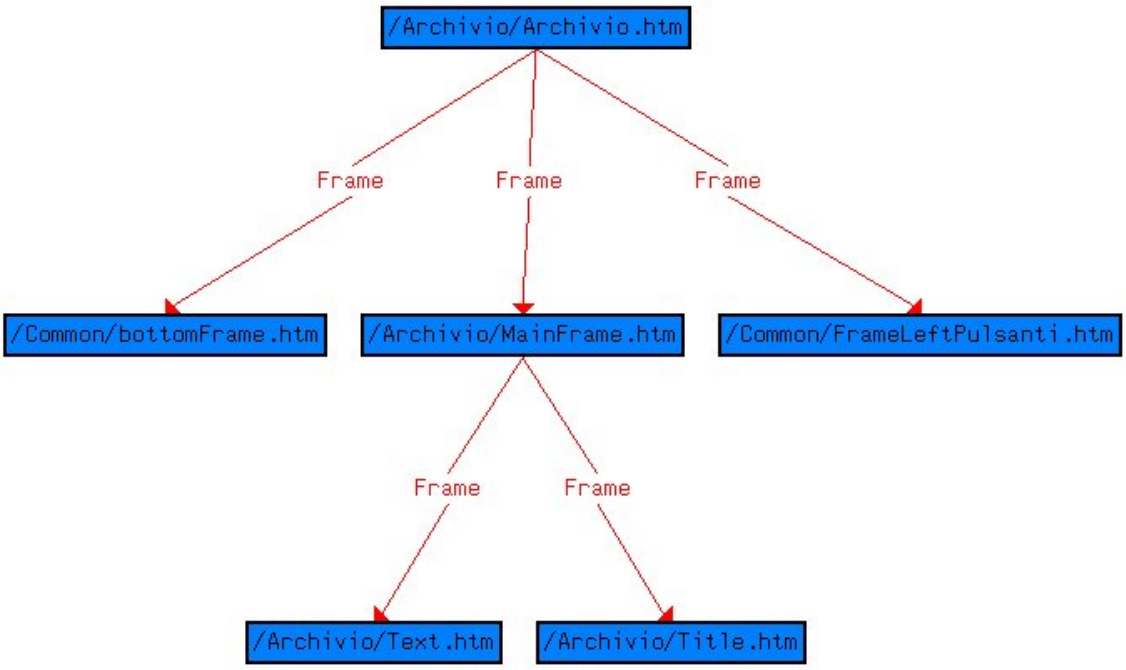


Figura 7 – Diagramma dei collegamenti a partire da Archivio.htm

Anche questa sottoparte dell'applicazione web è candidata ad assolvere ad una funzionalità dell'applicazione web. C'è da notare che nel diagramma non si sono considerati parecchi oggetti scaricabili, anch'essi residenti nella stessa cartella Archivio ed immagini, che si trovano invece nella cartella images.

Cartella Cerca

Si può vedere come esista ancora la stessa organizzazione in frame, con le pagine Mainframe, Title e Text, per cui si omette il grafo. Piuttosto può essere interessante lo studio della pagina text.htm, la quale contiene una form con dei parametri e un blocco javascript. I parametri della form sono i seguenti:

Tipo	Nome
select	engine
text	terms
select	op
button	

La form non ha alcun campo action, quindi non c'è alcun meccanismo esplicito di invio di parametri (ciò non toglie che vi sia un meccanismo parametrico, codificato in Javascript).

Cartella Forum

Anche per questa cartella si inizia dalla pagina MainFrame.htm, avendo ormai riconosciuto una convenzione fissa per i frame generali.

Figura 8 – Pagine collegate a MainFrame.htm

File MainFrame.htm

Si vede una situazione diversa: la pagina principale di questa cartella non si trova nella cartella Forum, ma nella cartella NT/Bulletin. Il motivo di questa situazione è semplice: questa sottoparte dell'applicazione deve contenere delle interazioni lato server. Siccome lo sviluppatore ha indicato come tecnologia scelta per queste funzionalità ASP, linguaggio che gira su server Windows NT, si può presumere che la sottoparte attuale dell'applicazione web comprenda anche la cartella NT/Bulletin che si va ora ad analizzare a partire da default.asp.

In questo grafo si vede per la prima volta il simbolo di rombo rosso, che indica pagine server ASP.

File default.asp

La pagina default.asp si limita a costruire una pagina client (cui si dà lo stesso nome per convenzione), si riporta direttamente il grafo partente da quest'ultima. Le pagine client costruita verranno rappresentate dal trapezio celeste.

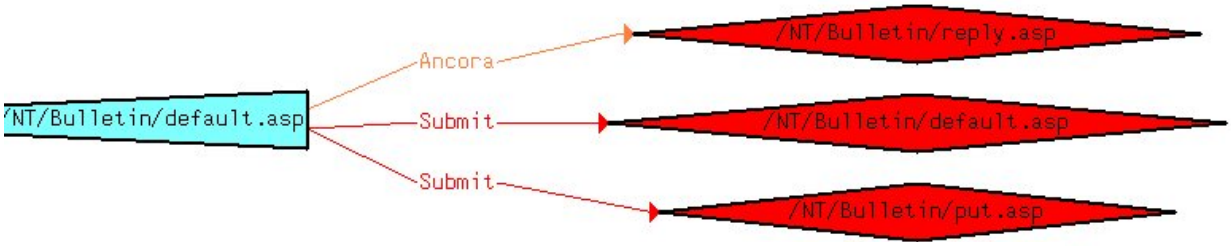


Figura 9 – Pagine collegate alla pagina client costruita default.asp

Dall'analisi della pagina server si vedono anche altri elementi: prima di tutto esistono molti blocchi VBScript, uno dei quali istanzia anche un oggetto interfaccia di tipo ADODB.RecordSet e interagisce con i suoi metodi.

Analizzando invece la pagina client costruita, si trovano due form, la prima con un parametro nascosto e il pulsante submit, l'altra con i seguenti parametri:

Tipo	Nome
Text	Name
Text	Email
Text	Title
textarea	Body
image	submit

L'analisi prosegue ora con reply.asp e put.asp.

File reply.asp

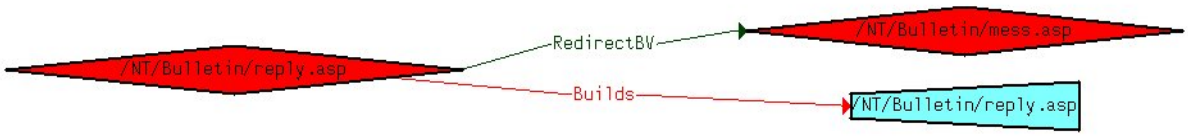


Figura 10 – Pagine collegate a reply.asp

Questa pagina si collega con mess.asp e con la pagina client che costruisce. Nella pagina server ci sono dei blocchi VBScript. Nella pagina client c'è un blocco javascript, un'ancora ad un oggetto di tipo mailto ma parametrizzato da una variabile server e una form con i seguenti parametri:

Tipo	Nome
Text	Name
Text	Email

Text	Title
textarea	Body
image	Submit

File mess.asp

Anche questa pagina server costruisce una pagina client. La pagina server contiene un blocco VBScript e interfaccia un oggetto CDONTS.Newmail.

File put.asp

Questa pagina server non crea una pagina client e si rilevano solo due blocchi VBScript.

Con questa pagina si è conclusa l'analisi delle pagine raggiungibili in questa sotto parte dell'applicazione web.

C'è però ancora un file, tool.asp, che si trova in questa cartella ma non risulta connesso ad alcun file. Ciò non significa che questo file (che tra l'altro crea anche un oggetto ADODB.Connection per l'interfaccia con un database) non faccia parte dell'applicazione, ma semplicemente che sia collegata ad essa con un meccanismo che esula da quelli analizzati automaticamente.

Anche in questo caso si può arrivare alla conclusione che le pagine contenute nelle due cartelle Forum e Bulletin contribuiscano ad uno stesso caso d'uso.

Per compendiare tutte le informazioni raccolte su questa sottoparte dell'applicazione web, si dovrebbe tracciare il diagramma dei collegamenti a partire da Forum.htm (oppure più semplicemente da default.asp). Trattandosi però di una parte dell'applicazione web attiva, si preferisce la rappresentazione con un class diagram, nel quale vengono indicate anche le interazioni con le interfacce, i form ed i parametri di questi ultimi.

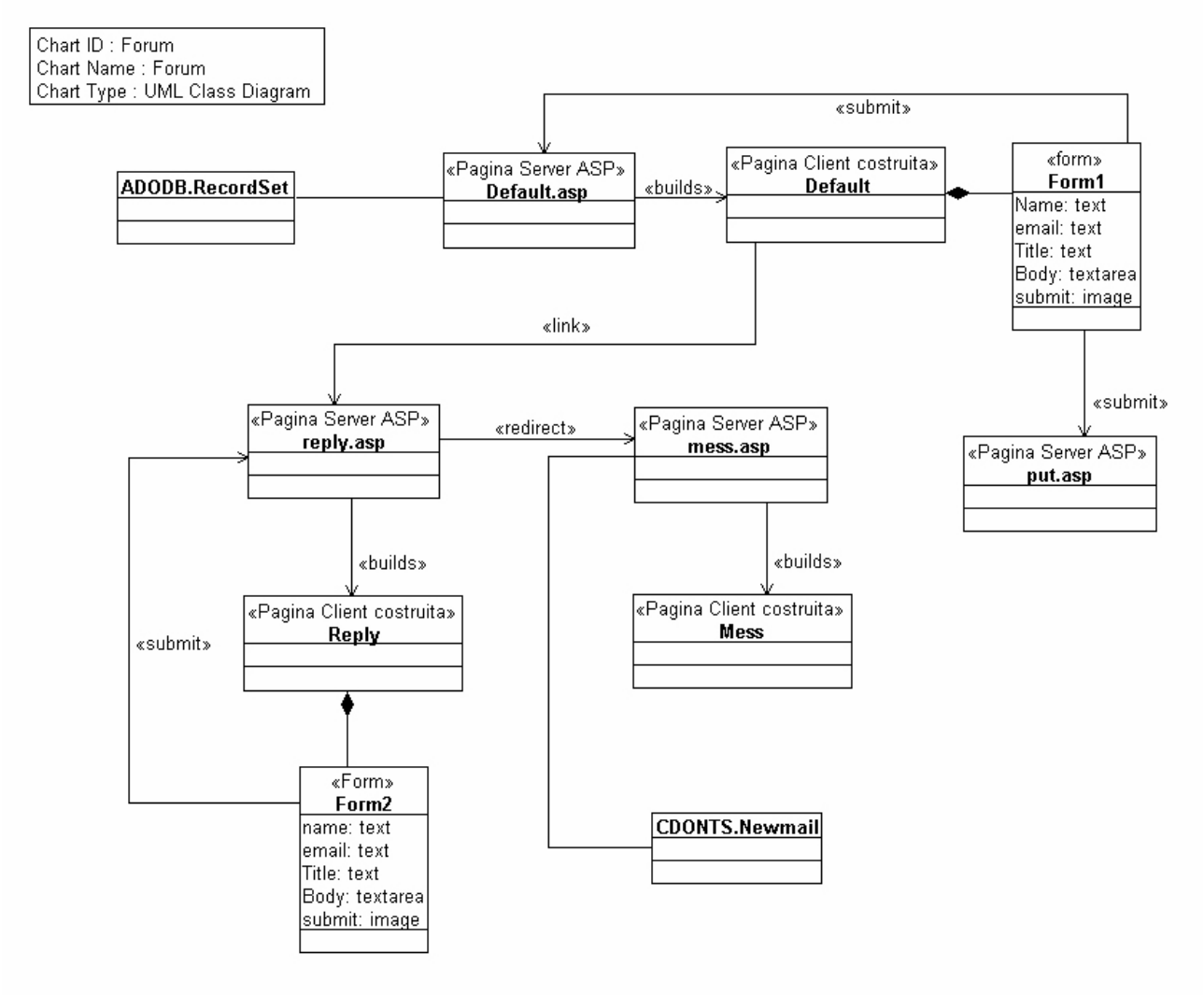


Figura 11 – Class diagram delle pagine della cartella Bulletin

Cartella Caso

Anche in questa cartella viene utilizzata la solita struttura di frame, quindi l'analisi passa direttamente a text.htm.

File text.htm

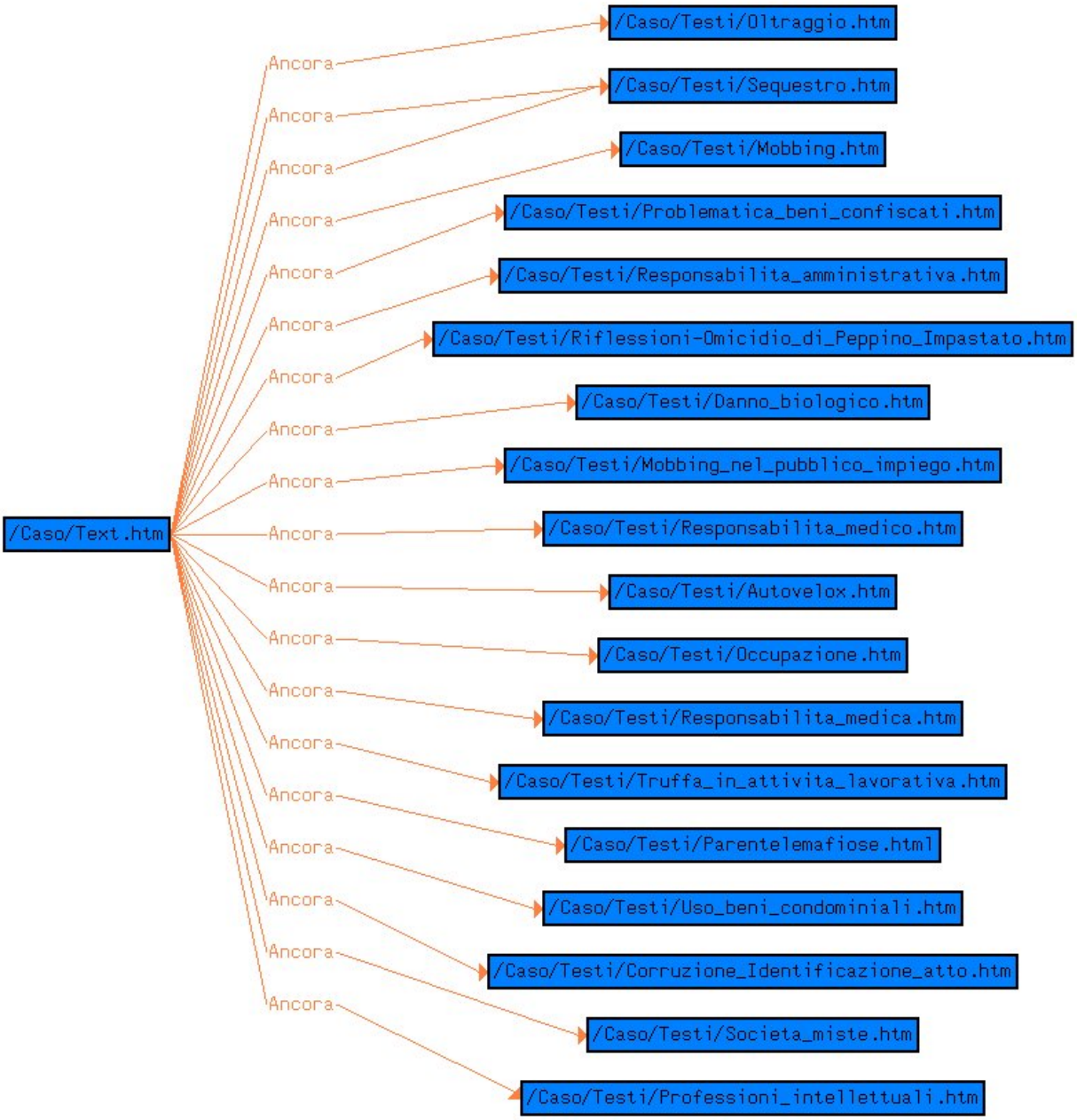


Figura 12 – Pagine collegate a Text.htm

Questa pagina, come si può vedere dal grafo, permette di raggiungere tutta una serie di altre pagine HTML. Analizzando queste ultime si vede come esse contengano riferimenti a oggetti scaricabili (file in formato zip) contenuti nella cartella archivio e ancore ad altre pagine contenute in questo stesso elenco.

In conclusione si può affermare che questa sottoparte dell'applicazione web potrebbere costituire un caso d'uso indipendente, a meno del legame vigente con il caso d'uso della cartella archivio. Confrontando con la versione on-line del sito si può vedere come i collegamenti con l'archivio siano solo accessori alla navigazione. Infatti gli argomenti trattati dai file scaricabili sono approfondimenti della questione trattata nella pagina.

Cartella Specialisti

Anche in questo caso è utilizzato il solito sistema di frame, per cui si comincia l'analisi direttamente da text.htm

File text.htm

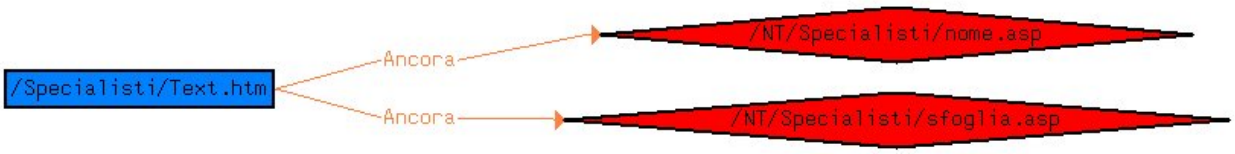


Figura 13 – Pagine collegate a Text.htm

Da questa pagina HTML è possibile raggiungere due pagine server, nome.asp e sfoglia.asp. Tali pagine saranno i prossimi oggetti dell'analisi, anche se si trovano in un'altra cartella, NT/Specialisti. Per questa cartella si può ripetere lo stesso discorso fatto per la cartella Bulletin nello studio della cartella Forum, che porta alla conclusione di unificare lo studio delle due cartelle NT/Specialisti e Specialisti.

File nome.asp

Qui si verifica una situazione curiosa: nome.asp non esiste nel codice sorgente. Si suppone, quindi, che tale pagina manchi a causa di un errore. Una conferma a questa supposizione giunge dal fatto che esiste una pagina nome2.asp. Probabilmente, si deve essere verificata una situazione nella quale sia stato aggiornato il nome della pagina ma non quello del collegamento o viceversa. Si analizza pertanto la pagina nome2.asp in vece di nome.asp.

La pagina server costruisce una pagina client, contiene numerosi blocchi VBScript e istanzia tre oggetti interfaccia: ADODB.RecordSet, ADODB.Connection e CDONTS.Newmail.

La pagina client costruita contiene invece una form con numerosi parametri:

Tipo	Nome
text	Nome
text	Cognome
select	Professione
text	Posta
text	Citta
select	Provincia
text	Ufficio
text	Ufficio2
text	Telefono
text	Telefono2
text	Cellulare

textarea	Note
select	GiornoForum
select	OraForum
select	GiuridicaCome
textarea	Suggerimenti
submit	Invia
reset	Reset

Questa form rimanda, via submit, alla pagina nome.asp stessa.

File sfoglia.asp

Questa pagina server costruisce una pagina client, ha dei blocchi VBScript e istanzia due oggetti interfaccia di tipo ADODB.RecordSet e ADODB.Connection.

La pagina client costruita contiene una form con i seguenti parametri:

Tipo	Nome
text	Cognome
select	Professione
text	Citta
select	Provincia
submit	Invia
reset	Reset

All'atto del submit, i risultati della form vengono inviati alla pagina sfoglia.asp stessa.

A questo punto non vi sono più componenti collegate da analizzare, ma ci sono ancora pagine attive nella cartella, admin.asp e aggiusta.asp. Questa volta, però i nomi delle pagine attive suggeriscono che esse possano essere punti di partenza di altri percorsi di navigazione, riservati stavolta ad una categoria di utenti diversa.

File admin.asp

Questa pagina server non sembra collegata ad alcun altra, per cui potrebbe assolvere da sola ad un caso d'uso. Essa costruisce una pagina client, istanzia due oggetti ADODB.RecordSet e ADODB.Connection e ha dei blocchi VBScript. La pagina client invece contiene una form, i cui parametri sono:

Tipo	Nome
text	Nome
text	Cognome
select	Professione
text	Posta
text	Citta
select	Provincia
text	Ufficio
text	Ufficio2

text	Telefono
text	Telefono2
text	Cellulare
textarea	Note
select	GiornoForum
select	OraForum
select	GiuridеaCome
textarea	Suggerimenti
button	Invia
reset	Reset

File aggiusta.asp

Questa pagina server ha dei blocchi VBScript, istanzia un oggetto ADODB.RecordSet, ma non sembra avere alcun legame con altre pagine della sottoparte in analisi, né costruisce pagine client. Si può pensare che questa pagina esegua una routine, richiamata dall'amministratore o da qualche altra pagina, ma in un modo che non è tra quelli contemplati.

In conclusione, dall'analisi di queste due sottocartelle si può ipotizzare che corrispondano ad un caso d'uso del sistema, suddiviso però in più funzionalità, le quali possono essere realizzate da diverse tipologie di attori del sistema.
Anche in questo caso, si preferisce visualizzare il class diagram ricavabile da queste informazioni piuttosto che il diagramma dei collegamenti.

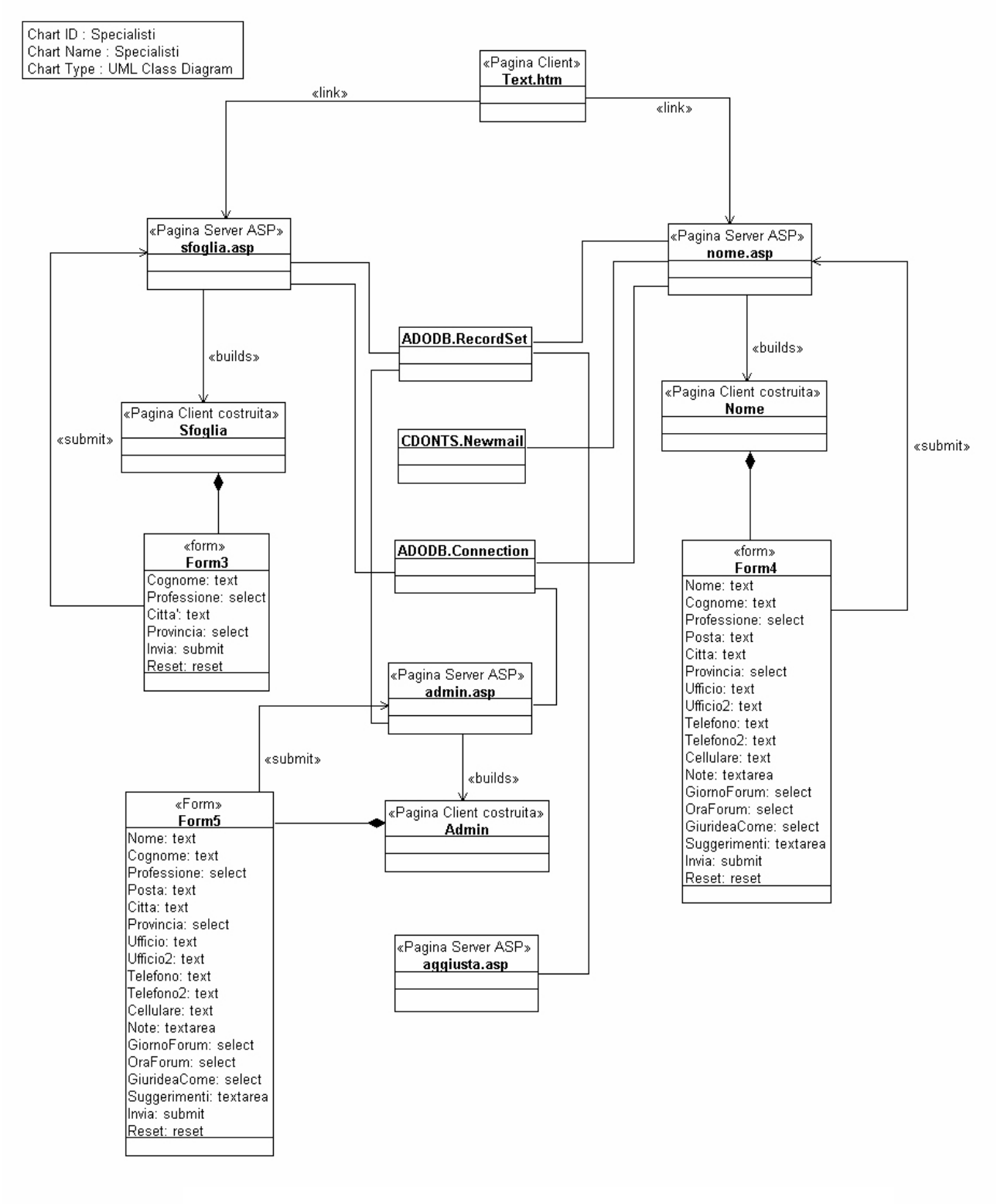


Figura 14 – Class diagram risultante dallo studio dei file della cartella Specialisti

Cartella ChiSiamo

Anche quest'ultima cartella è conforme alle precedenti nell'uso dei frame.

File text.htm

Questa pagina è molto semplice, contiene solo due ancore ad indirizzo di posta elettronica.

Questa sottoparte dell'applicazione web è già conclusa, dal nome si può pensare ragionevolmente che rappresenti la pagina di riferimento dello sviluppatore.

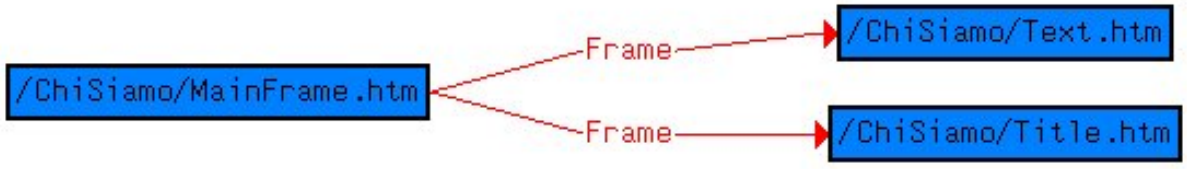


Figura 15 – Pagine collegate a MainFrame.htm

Altre cartelle

A questo punto si può considerare esaurita la zona connessa dell'applicazione web, limitatamente almeno ai collegamenti presi in esame. Leggendo la directory dell'applicazione web sorgente si trovano invece alcune cartelle che non sono state analizzate ancora. La cartella common contiene due file di utilizzo comune, richiamati come bordi delle pagine con frame. La cartella images contiene unicamente immagini, quindi può essere trascurata ai fini dello studio funzionale dell' applicazione web. Rimane così una sola cartella rilevante, NT/Mailing, che si va ora ad analizzare.

In questa cartella ci sono due file, login.asp e mailing.asp. Si ricavano i due grafi seguenti:

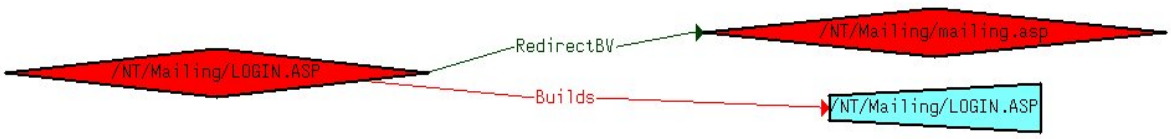


Figura 16 – Pagine collegate a Login.asp

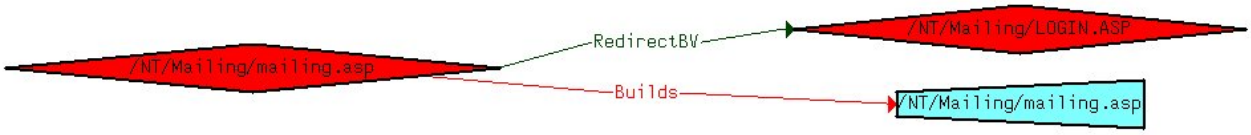


Figura 16 – Pagine collegate a mailing.asp

Come si può vedere, le due pagine sono interconnesse tra di loro.

La pagina login.asp crea una pagina client, contiene dei blocchi VBScript e utilizza due oggetti interfaccia di tipo ADODB.RecordSet e ADODB.Connection.

La pagina client costruita da login.asp contiene un form che invia il submit alla stessa pagina server da cui è stata generata, con i seguenti campi:

Tipo	Nome
text	UserID
password	codice
submit	

La pagina Mailing.asp si comporta in maniera simile: costruisce una pagina client, contiene dei blocchi VBScript e utilizza interfacce di tre tipi: ADODB.RecordSet, ADODB.Connection e CDONTS.NewMail.

La pagina client costruita ha un form con i seguenti campi:

Tipo	Nome
text	Titolo
textarea	Messaggio
submit	Invia
reset	Reset

Le informazioni trovate si riassumono nel class diagram della figura 17.

In conclusione si può pensare che i file di questa cartella contribuiscano alla realizzazione di un ulteriore caso d'uso, ma che siano momentaneamente irraggiungibili dal resto dell'applicazione web, forse perché tale funzione non è stata sviluppata completamente, oppure perché è riservata ad utenti particolari.

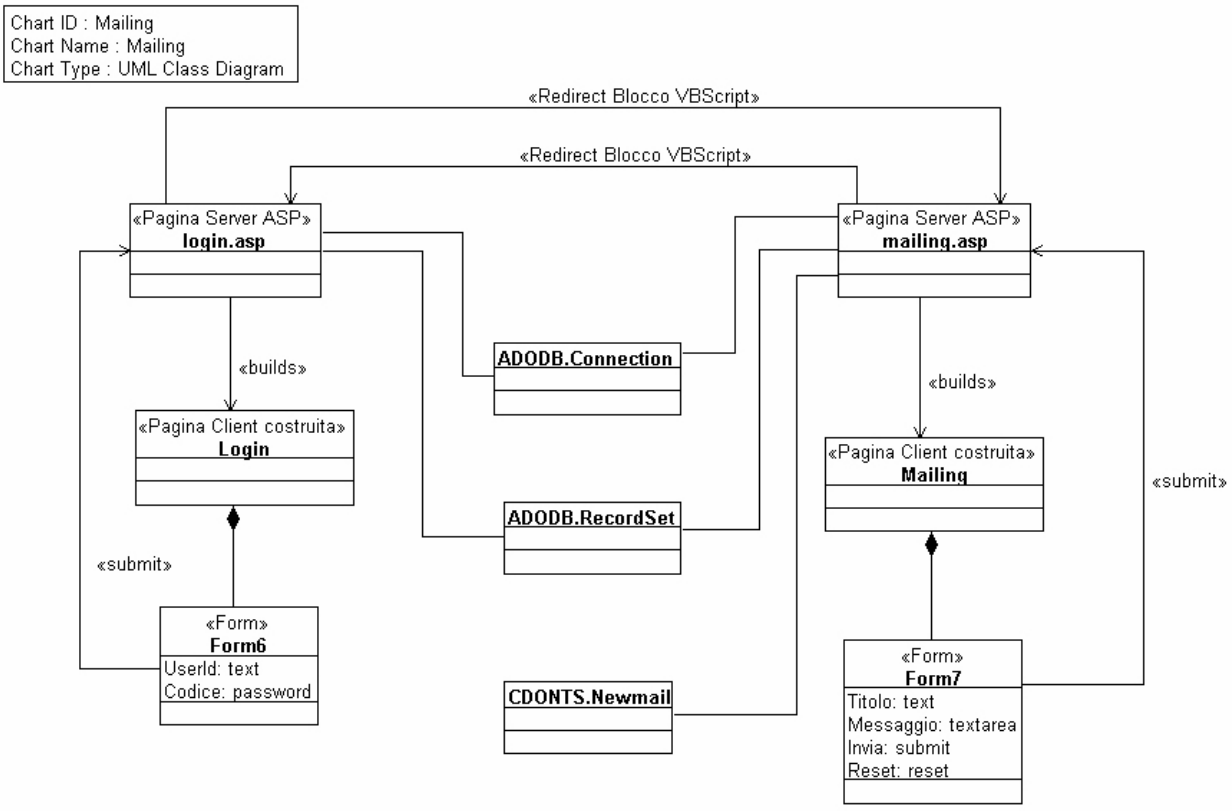


Figura 17 – Class diagram risultante dall'analisi della cartella mailing

2.3 Conclusioni

L'applicazione web Giuridea è strutturato in maniera semplice e schematica. Esiste infatti un meccanismo di suddivisione in frame della finestra comune per ognuna delle funzionalità dell'applicazione web. Inoltre le pagine e gli oggetti principali relativi ad ogni funzionalità sono contenuti in una cartella, il cui nome richiama il nome della funzionalità, così come compare come etichetta sui collegamenti.

Eccezioni sono rappresentate dalle immagini, le quali si trovano tutte raggruppate in una cartella images (si tratta comunque di una procedura largamente utilizzata dagli sviluppatori) e dalle pagine contenute nella cartella Common, che sono effettivamente comuni a tutte le funzionalità. Ci sono poi le cartelle contenute nella cartella NT, le quali contengono le funzionalità svolte in ASP, distinguendole da quelle svolte con il sistema operativo Unix. Comunque, basta accoppiare a due a due cartelle sotto Unix con cartelle sotto NT per riottenere le funzionalità.

Questa armonia nella struttura del sito permette alla metodologia di reverse engineering di essere più efficace, in quanto è possibile una decomposizione dell'applicazione web in componenti molto piccole.

Un altro fattore che ha contribuito alla semplicità dell'analisi è dato dall'utilizzo dei frame. Infatti è noto come un sito senza frame abbia molti più collegamenti di un sito con frame per cui il grafo corrispondente è molto più ingarbugliato e diventa molto più difficile riconoscere le componenti funzionali dell'applicazione web.

Si può anche affermare che un'applicazione web così strutturata possa essere facilmente estesa, con l'aggiunta di nuove funzionalità, almeno dal punto di vista dell'inserimento di nuove componenti indipendenti. Nulla si può dire, con questa analisi, sulla condivisione dei database tra le varie funzionalità, e quindi sulla loro effettiva indipendenza.

Dallo studio condotto sul caso di studio, si possono infine ipotizzare alcuni casi d'uso. Tali ipotesi potranno costituire un'utile guida per chi si appresti alla comprensione dell'applicazione web. Ecco un elenco di questi candidati casi d'uso:

- Indice, che funge da punto di partenza per la navigazione;
- Novità, da cui si accede alle ultime notizie e/o iniziative del sito;
- Archivio, da cui si può accedere ai documenti bibliografici relativi ai casi trattati;
- Cerca, che permette di trovare uno specialista dal database apposito;
- Forum, da cui si può dialogare, ad orari prefissati, con gli specialisti che collaborano al sito;

- Caso, nel quale sono riportati i casi giudiziari affrontati;
- Specialisti, con la quale le persone interessate a fornire consulenze per conto del sito web possono iscriversi;
- Chi siamo, dal quale si accede agli indirizzi cui sono reperibili i responsabili del sito web;
- Mailing List, col quale il webmaster può inviare dei messaggi di posta elettronica a tutti gli specialisti iscritti.

Questi casi d'uso sono stati ipotizzati a partire dalla suddivisione che è stata riscontrata nello studio dell'applicazione web e dalla navigazione interattiva che è stata fatta sulla versione on line del sito. Molto utile, a tal fine, è stata anche l'analisi della semantica dei nomi di file, che è stata sempre coerente con l'effettivo utilizzo dei file stessi.

In conclusione, alcune informazioni dimensionali sull'applicazione web analizzata: essa è formata da 201 file, suddivisi in 19 cartelle, per un totale di 4.26 MByte. Di questi file 170 si trovano sul web server Unix, mentre i rimanenti 31 file si trovano sul web server Windows NT. Sono state analizzate 19 pagine server e 55 pagine client.

3 Caso di studio: DIS

3.1 Analisi preliminari

Il secondo caso di studio riguarda l'applicazione web del Dipartimento di Informatica e Sistemistica – DIS. Quest'applicazione web, pubblicata all'indirizzo www.dis.unina.it si propone come punto di accesso a tutta una serie di informazioni e servizi rigurdanti il dipartimento, come le informazioni riguardanti l'organico, le pubblicazioni, le tesi, gli orari delle lezioni, il materiale didattico, etc. Recentemente questo sito è stato rinnovato, con il concorso di alcuni studenti del corso di Reti di Calcolatori del 2000/2001 tenuto dal prof. Ventre, con l'aggiunta di pagine server, realizzate con tecnologia ASP, in modo da rendere l'aggiornamento del sito automatico con l'aggiornamento di un database.

Da una prima analisi, risulta che tutta l'applicazione web si trova nella cartella iniziale (ad eccezioni delle immagini che si trovano nella cartella images, che però non sono rilevanti ai fini dello studio). L'unica possibilità consiste nell'iniziare l'analisi dalla pagina iniziale, index.htm, la quale è il punto d'accesso per l'utente generico.

3.2 Analisi statica

File index.htm

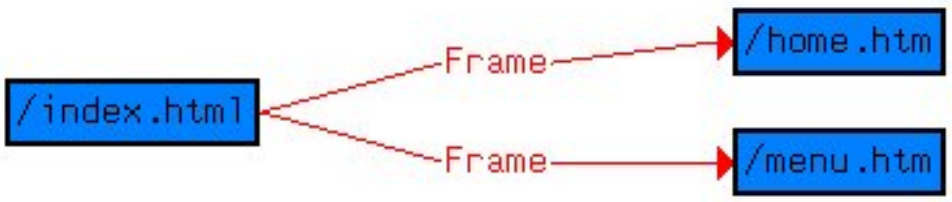


Figura 18 – Pagine collegate a index.html

La pagina index funge da contenitore per i due frame home.htm e menu.htm.

File home.htm

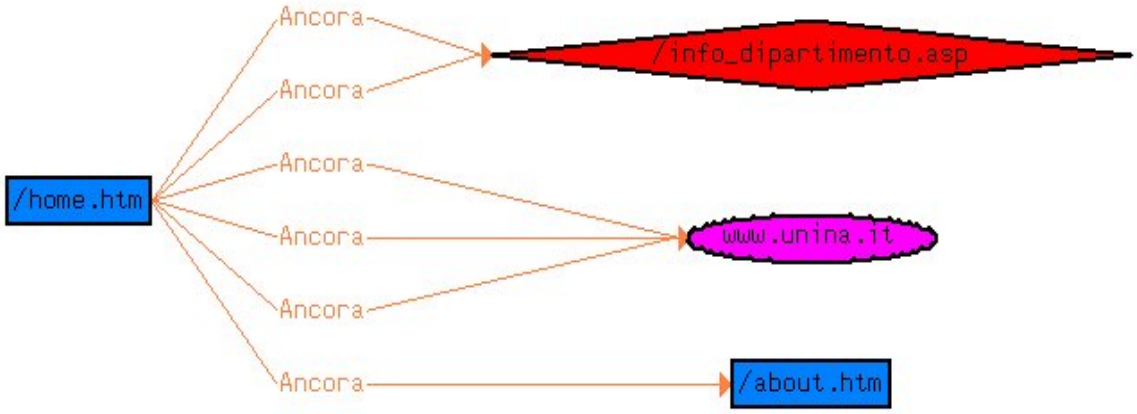


Figura 19 – Pagine collegate a home.htm

Questa pagina, che funge da copertina per il sito, contiene un collegamento ipertestuale alla pagina about, la quale ha però solo un collegamento a sé stessa e permette di effettuare il download di alcuni file presenti sul sito.

La pagina server info_dipartimento.asp crea una pagina client, e si interfaccia con un database con un oggetto ADODB.RecordSet.

In conclusione, si può affermare che la sottoparte dell'applicazione web che inizia con home.htm realizza una funzionalità, precisamente quella di fornire informazioni generali sul dipartimento e sugli sviluppatori dell'applicazione web. Nonostante ciò, è utilizzato già in questo caso un meccanismo server di accesso al database per ricavare le poche informazioni necessarie a queste pagine.

File menu.htm

Questa pagina funge da introduzione al resto dell'applicazione web. Dal suo

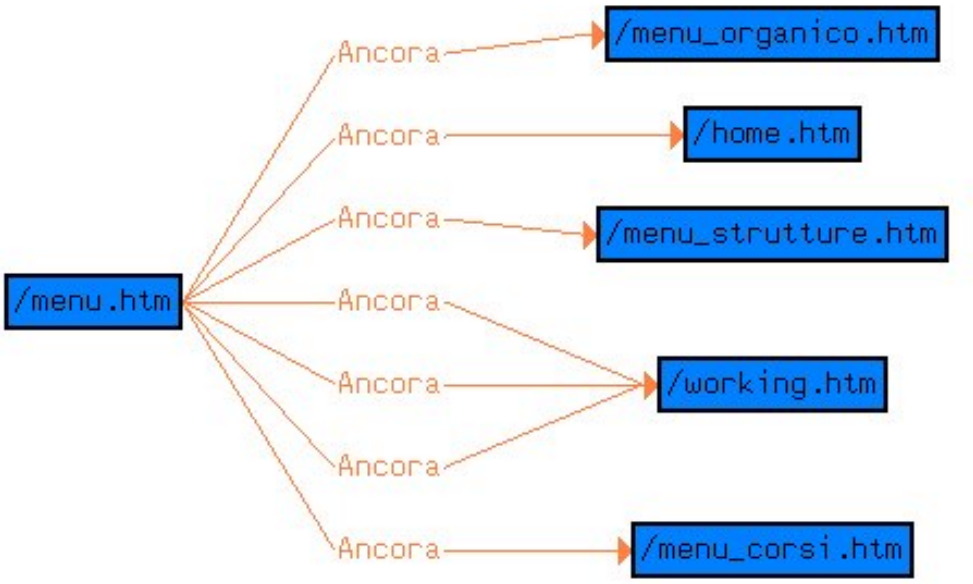


Figura 20 – Pagine collegate a menu.htm

diagramma dei collegamenti si possono vedere le pagine staticamente collegate. Tra esse si ritrova home.htm che è stata già analizzata. Si nota poi che working.htm ha ben tre collegamenti. Andandola ad analizzare si vede come questa pagina non abbia alcun collegamento uscente, per cui si può presumere che si limiti ad indicare il fatto che la funzionalità in questione non sia stata ancora implementata.

File menu_organico.htm

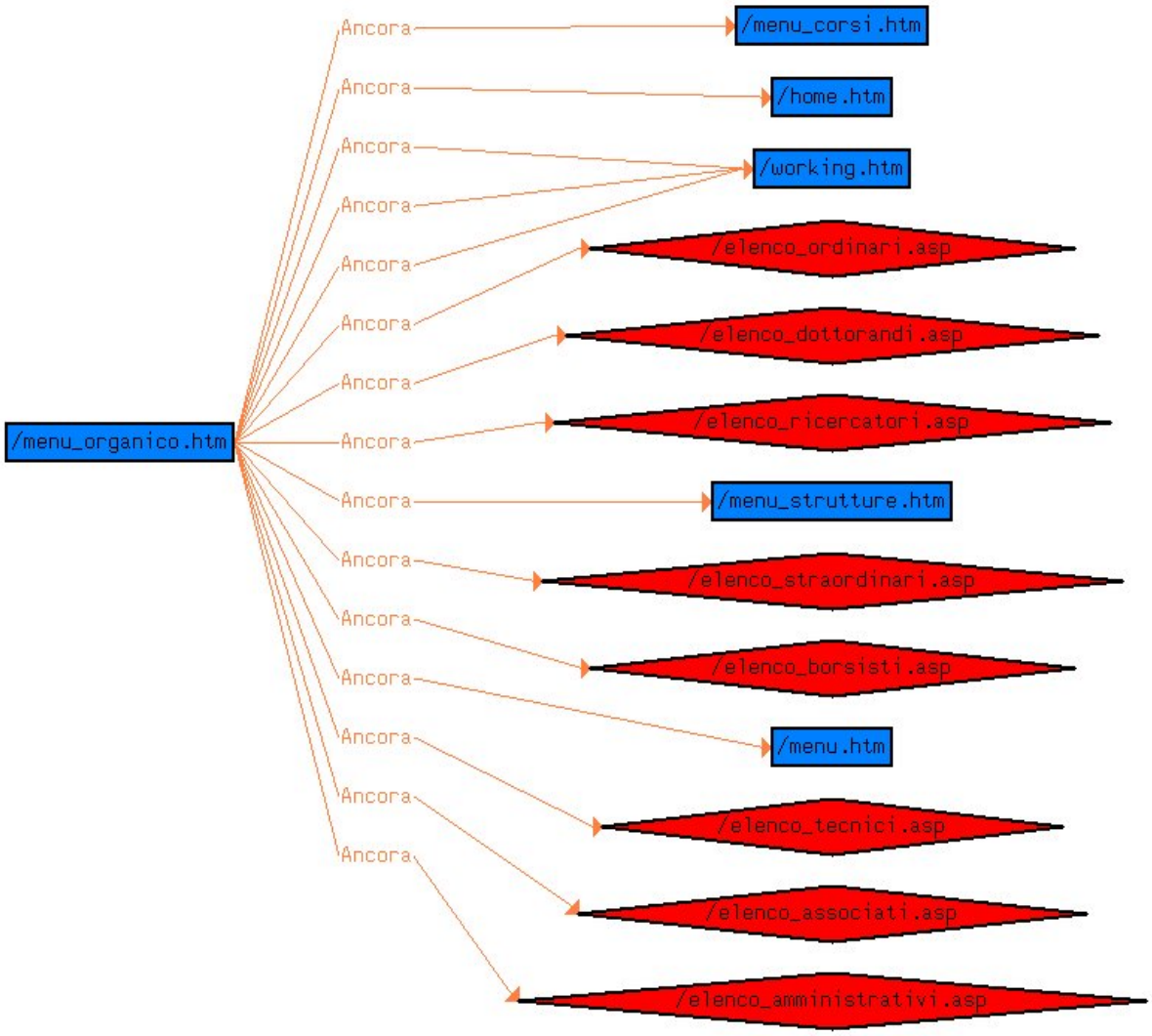


Figura 21 – Pagine collegate a menu_organico.htm

Come si può vedere, molte pagine sono collegate a menu_organico.htm. Le pagine HTML, però, o sono pagine già analizzate, oppure sono tra quelle che ci si propone di analizzare, per cui ci si limita all'analisi delle pagine server.

File elenco_ordinari.asp

Analizzando questa pagina, si può vedere come essa non abbia collegamenti uscenti, ma costruisca una pagina client.

La pagina server istanzaia un oggetto per l'interfaccia a database, di tipo ADODB.RecordSet ed ha numerosi blocchi VBScript. La pagina client contiene poi

un ancora ad un indirizzo di posta elettronica, il cui valore però dipende da variabili server.

Figura 22 – Pagine collegate a elenco_ordinari.asp

Si può concludere che questa pagina, da sola, riesca ad assolvere ad una sottofunzionalità, come sembrerebbe indicare anche il suo nome. Dagli elementi visti, si può ipotizzare che l'accesso al database sia in lettura e non parametrico, in quanto non si sono intercettati form che permettano di inserire informazioni.

File elenco_dottorandi.asp

Già dal nome di questo file, si può ipotizzare come svolga una funzionalità simile al precedente. L'ipotesi trova conferma dall'analisi della pagina, in quanto essa contiene oggetti dello stesso tipo di elenco_ordinari.asp, e lo stesso vale anche per la pagina client che viene costruita.

Altri file del tipo elenco_*.asp

Tutte le pagine asp che si incontrano hanno esattamente la stessa fisionomia, almeno a livello dell'analisi svolta.

File menu_strutture.htm

Questa pagina ha una struttura che ricorda quella di menu_organico.htm. Infatti si può vedere come si colleghi a pagine già note, oppure da analizzare, e anche alla pagina server elenco_strutture.asp.

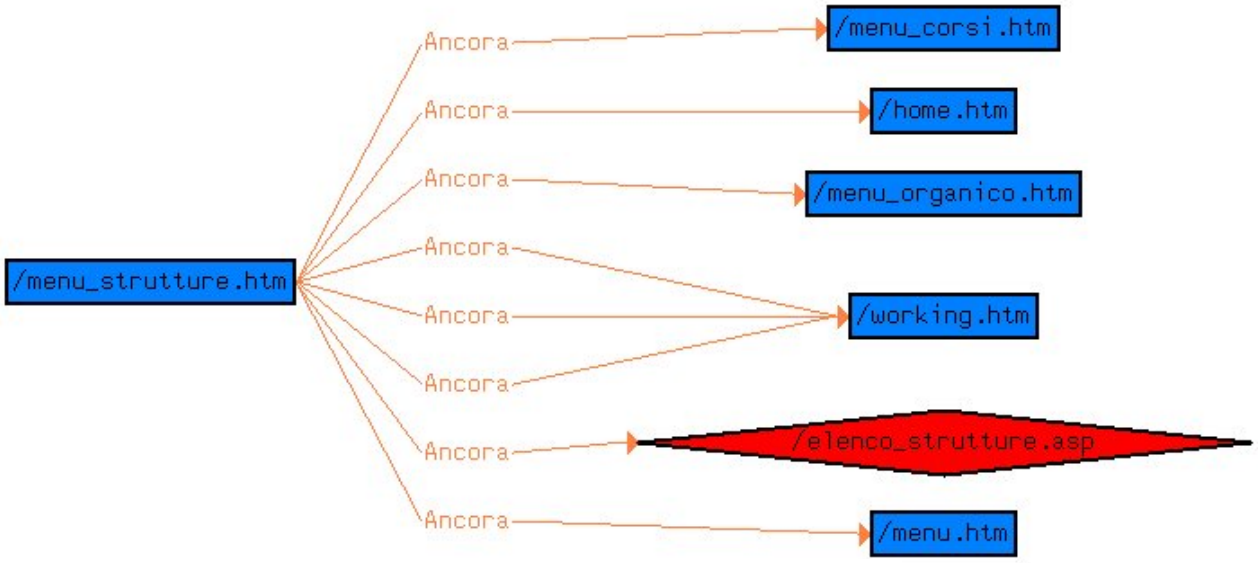


Figura 23 – Pagine collegate a menu_strutture.htm

File elenco_strutture.asp

Questa pagina server ha una struttura che ricorda molto da vicino le analoghe pagine collegate a menu_organico.htm. Infatti, costruisce una pagina client, ha blocchi VBScript e utilizza un oggetto interfaccia ADODB.Recordset. La pagina client costruita stavolta non ha alcun collegamento, nemmeno verso indirizzi di posta

Figura 24 – Pagine collegate a elenco_strutture.asp

elettronica.

Si può subito concludere come la sottoparte dell'applicazione web con menu_strutture.htm realizzi una sottofunzionalità, analoga a quella realizzata da menu_organico.htm.

File menu_corsi.htm

Anche questa pagina rispecchia la struttura delle altre due omologhe viste in precedenza, per cui l'analisi passa immediatamente a elenco_insegnamenti.asp.

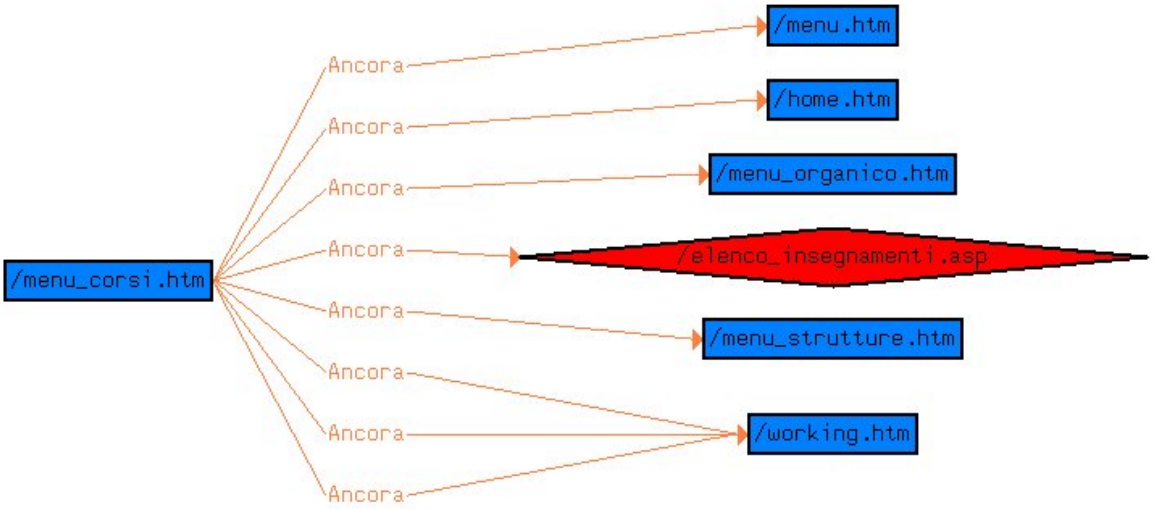


Figura 25 – Pagine collegate a menu_corsi.htm

File elenco_insegnamenti.asp

Questa pagina server contiene, come le altre pagine server viste, dei blocchi

Figura 26 – Pagine collegate a elenco_insegnamenti.asp

Figura 27 – Pagine collegate a elenco_insegnamenti.asp

VBScript e un oggetto interfaccia di tipo ADODB.Recordset. Costruisce anche una pagina client, la quale ha però una particolarità: contiene infatti un ancora alla pagina info_corsi.asp, con dei parametri dipendenti da variabili server.

La pagina server info_corsi.asp costruisce anch'essa una pagina client, contine numerosi blocchi VBScript ed utilizza ben due oggetti di tipo ADODB.RecordSet.
La pagina client costruita contiene anche un'ancora ad un indirizzo di posta elettronica, dipendente però da variabili server.

Con questa pagina, si può considerare conclusa l'analisi statica dell'applicazione web, in quanto si può vedere come non vi siano altre pagine non raggiungibili.
A questo punto si possono raccogliere tutte le informazioni in un class diagram.

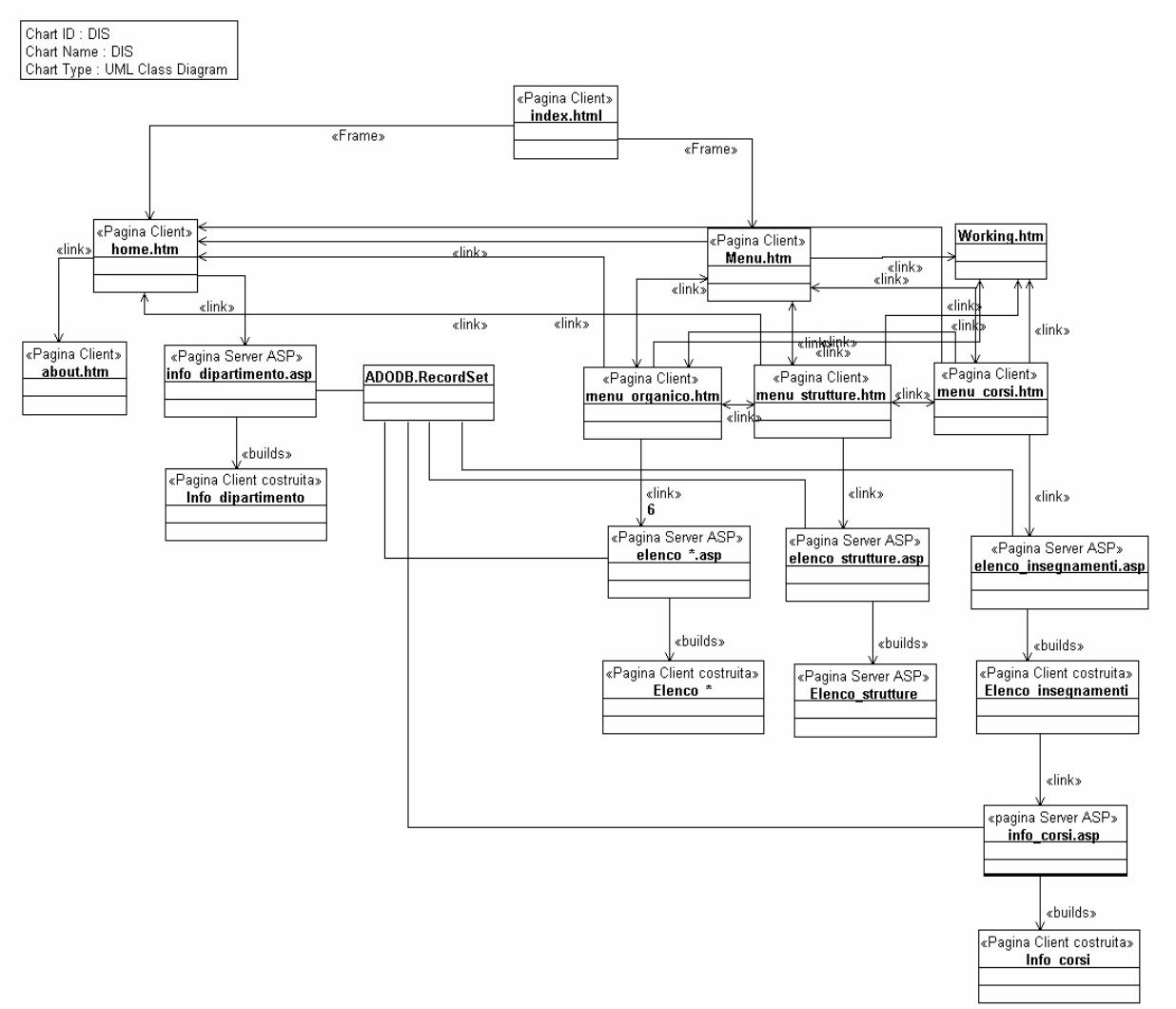


Figura 28 – Class diagram risultante dall'analisi del sito web del DIS

In questo class diagram si sono utilizzate, per chiarezza espositiva, due convenzioni: nel caso in cui tra due pagine esistono due link opposti, si visualizza una sola linea, con frecce ad entrambe le estremità; nel caso delle 6 pagine asp raggiungibili da menu_organico, si è scelto di indicarne una sola generica con molteplicità 6.

3.3 Conclusioni

Quest'applicazione web é molto piccola, anche in confronto al sito Giuridea esaminato in precedenza. Questo perché è ancora allo stato prototipale e non ancora rilasciata. Questa situazione trova conferma nella circostanza che buona parte delle funzionalità previste non sono disponibili nella versione analizzata.

L'organizzazione del sito è ancora una volta quella a frame. Inoltre si è utilizzato un meccanismo con il quale i collegamenti principali sono sempre visualizzati nel frame sinistro, mentre le pagine da consultare sono sempre nel frame centrale.

Un limite all'estensione di quest'applicazione web è dato dall'assenza di sottodirectory, ad eccezione di quella dedicata alle immagini. L'analisi dell'applicazione web sarebbe diventata molto più difficile, qualora esso avesse avuto più pagine, in quanto non si può pervenire ad una distinzione funzionale immediatamente.

Un altro limite è dato dal fatto che esiste un unico database, cui si appoggiano tutte le funzionalità. L'aggiunta di altre funzionalità deve rispettare la struttura di tale database.

I casi d'uso che si possono ipotizzare dallo studio di quest'applicazione web sono i seguenti:

- Elenco ordinari
- Elenco dottorandi
- Elenco ricercatori
- Elenco straordinari
- Elenco borsisti
- Elenco tecnici
- Elenco associati
- Elenco amministrativi
- Elenco strutture
- Elenco insegnamenti

Tali casi d'uso corrispondono alle pagine server dell'applicazione. I nomi che sono stati dati ai casi d'uso coincidono con quelli delle pagine server che li realizzano.

In conclusione, alcune informazioni dimensionali sull'applicazione web analizzata: essa è formata da 28 file, compresi in due cartelle, per un totale di 2.58 Mbyte. 20 di

questi file sono pagine web (8 pagine client HTML e 12 pagine server ASP), e sono stati quindi analizzati dal sistema.

CAPITOLO 12 - ESTENSIONI FUTURE

Il lavoro contenuto in questa tesi può rappresentare l'origine di un progetto di più ampio respiro nell'ambito del reverse engineering di applicazioni web.
Affinchè ciò sia possibile, il lavoro deve essere espanso in varie direzioni:

- Estensione del campo di applicazione;
- Estensione degli strumenti automatici, fino alla copertura di tutta la metodologia;
- Estensioni teoriche della metodologia.

Per quanto riguarda l'estensione del campo di applicazione, bisogna operare prima di tutto in modo da arrivare al riconoscimento di ogni tag HTML (eventualmente anche quelli relativi alla visualizzazione), nonché ogni comando Javascript e VBScript. Tale lavoro richiede un notevole impegno di risorse umane. Di conseguenza sarebbe necessaria anche una estensione del progetto del database.
Sarebbe opportuna anche l'analisi di altri linguaggi. In particolare, durante il lavoro di tesi, ci sono state molte richieste riguardanti il linguaggio server PHP. E' possibile estendere il tool a tutti i linguaggi interpretati che utilizzano un modello architetturale client/server (vedi capitoli 2 e 3).

L'utilizzo di strumenti automatici che supportino il lavoro dell'analista in tutta la metodologia di reverse engineering teorizzata permetterebbe di poter analizzare applicazioni web di dimensioni ragguardevoli. Si è però visto come non sia possibile automatizzare completamente l'analisi, per cui si dovrebbero soltanto aggiungere degli strumenti che siano in grado di realizzare procedure di trattamento dell'informazione più complesse.
Scendendo nello specifico, il primo passo consiste nello sviluppo di uno strumento per l'astrazione delle informazioni estratte dal tool sviluppato in questa tesi. La realizzazione di questo tool può prendere avvio dalle considerazioni fatte sulla semantica del file taggato risultante dall'estrattore, contenuta nel capitolo 9.
Ulteriori strumenti di astrazione devono portare alla redazione automatica dei diagrammi considerati nella metodologia del capitolo 6 (diagramma dei collegamenti, class diagram, grafo di raggiungibilità). L'esperienza dell'analista deve poi portare alla definizione di metodi più utili di trattamento dell'informazione.
Queste estensioni sono contenute nella tesi riportata in bibliografia [1].
Ulteriori estensioni riguardano le altre due macrofasi della metodologia: analisi dinamica e analisi funzionale.

Per quanto riguarda l'analisi dinamica, l'approccio dovrà essere profondamente diverso, in quanto un eventuale tool automatico dovrebbe andarsi ad integrare con web server e browser al fine di poter seguire e tracciare l'esecuzione dell'applicazione web. Una difficoltà che dovrà superare lo sviluppatore di tali tool sarà legata alla sovrapposizione degli effetti dovuti a web server e browser.
L'analisi funzionale sarà sicuramente la fase meno automatizzabile del processo, per cui si può pensare solo a tool automatici di trattamento delle informazioni, le quali dovranno però essere ricavate in base alla conoscenza ed esperienza dell'ingegnere del software che effettua la comprensione.
Potrebbe essere interessante integrare tali strumenti di reverse engineering nell'ambito di applicazioni orientate al progetto di applicazioni web, in modo da ottenere uno strumento di reengineering e manutenzione di un'applicazione web (in effetti l'idea iniziale che ha dato vita al progetto è stata l'osservazione delle utility fornite da Front Page e Dreamweaver, le quali sono però molto semplici e si limitano solo agli elementi statici di un'applicazione web).

Per quanto riguarda le estensioni teoriche, esse dovranno essere dettate dall'analisi di applicazioni web più complesse. In particolare, le regole che portano a scegliere l'ordine di analisi di un'applicazione web sono di tipo euristico e hanno dato buoni risultati nei casi di studio analizzati. Ciò non toglie che possano non essere le migliori in generale.
Lo studio delle macrofasi dell'analisi dinamica e funzionale dovrebbe essere meglio precisato, in funzione dei tool automatici di sviluppo che si andrebbe a progettare, aggiungendo, eventualmente, degli ulteriori diagrammi di rappresentazione.

CAPITOLO 13 - CONCLUSIONI

Il lavoro svolto in questa tesi si proponeva di raggiungere degli obiettivi teorici e pratici.

Tra gli obiettivi teorici, c'era l'esigenza di trovare delle metodologie e dei modelli formali che raggiungessero lo scopo di rappresentare applicazioni web.

Le ricerche condotte hanno portato a chiare conclusioni: questo ramo dell'ingegneria del software è in evoluzione, cosicché sono state trovate numerose metodologie di progetto, ma nessuna di queste sembra di applicazione generale. D'altronde il problema del progetto di un'applicazione web non può essere guidato solamente da considerazioni tecniche. Al contrario, prende sempre più piede l'idea che il progetto di un'applicazione web debba essere user-centered, orientato all'utente.

In mancanza di un accordo sulla scelta di una metodologia generale, si è cercato di adattare il formalismo UML alla rappresentazione delle applicazioni web, provandola su di un esempio e notando le incongruenze e le perdite di semantica.

Sulla scorta di quanto visto in fase di progettazione, si è rivolta l'attenzione verso il reverse engineering di un applicazione web. Non è stata trovata in letteratura alcuna metodologia a tal proposito, per cui si è cercato di proporne una originale.

Questa metodologia è stata ottenuta a partire dallo studio di un esempio, ed è stata consolidata, poi, provandola su due casi di studio reali. Nella definizione della metodologia si é cercato, ove opportuno, di alternare forme di rappresentazioni note (come i diagram di UML e il diagramma dei collegamenti, utilizzato anche da Conallen in [2]) ad altre più originali (come il grafo di raggiungibilità).

La metodologia di reverse engineering, descritta formalmente nel capitolo 6, rappresenta l'obiettivo teorico di questa tesi.

Gli obiettivi pratici della tesi sono legati all'automatizzazione di una parte della metodologia, precisamente l'estrazione di informazioni dal codice sorgente di un'applicazione web a supporto della sua comprensione.

A questo scopo si è proceduto ad uno studio delle informazioni da estrarre, finalizzato nella progettazione di un database. Si è, poi, definita la sintassi del file taggato che deve risultare dall'esecuzione dell'estrattore. Tutte le scelte fatte in queste ultime due fasi sono state tradotte in vincoli sul progetto dell'estrattore, per cui si è pervenuti ad una sua analisi dei requisiti precisa.

L'estrattore è stato, quindi, progettato seguendo la metodologia UML ed utilizzando automi a stati finiti per il riconoscimento dei tag e dei comandi presenti nel codice sorgente.
Si è cercata una conferma dell'utilità della metodologia di reverse engineering semiautomatizzata, provandola su due casi di studio, di media complessità. Effettivamente si sono ottenuti dei risultati abbastanza soddisfacenti, in quanto il processo di reverse engineering si è velocizzato notevolmente, una volta che l'estrattore ha automaticamente isolato le informazioni significative dalle altre.
L'analisi dei casi di studio ha portato anche ad esprimere alcune considerazioni di carattere generale sulla progettazione di applicazioni web. L'utilizzo dei frame rende la struttura dell'applicazione web molto più comprensibile, più vicina ad un albero che ad unn grafo connesso, permettendo una decomposizione funzionale dell'applicazione.

L'impressione generale che si trae è che, quando saranno disponibili tutte le estensioni previste nel capitolo precedente, dovrebbe essere possibile perseguire l'obiettivo generale, ovvero quello di procedere in maniera semiautomatica al reverse engineering di un'applicazione web.

Glossario

Access, software prodotto dalla Microsoft che permette di gestire database. Fa parte del pacchetto Microsoft Office, con gli altri componenti del quale si interfaccia completamente.

Apache Web Server, web server prodotto dalla Apache, e distribuito freeware. Può essere efficacemente utilizzato per interpretare, ad esempio, script CGI o Cold Fusion.

Applet, spezzone di un programma Java che viene eseguito sulla macchina client.

ASP, Active Server Pages, tecnologia per il lato server delle web application prodotta dalla Microsoft. Si caratterizza per la sua semplicità d'utilizzo e per l'interfacciabilità con gli altri componenti Microsoft.

Browser, software che permette di navigare nel World Wide Web, assumendo quindi il lato client di una web application.

Bytecode, codice intermedio, costruito compilando il codice sorgente di un programma scritto in linguaggio Java. Tale codice intermedio verrà poi interpretato dalla Java Virtual Machine, che può costituire un'estensione del browser.

CFML, Cold Fusion Markup Language, linguaggio di script utilizzato dalla tecnologia ColdFusion

CGI, Common Gateway Interface, tecnologia che peremette un'interazione client-server, concentrando tutte le azioni elaborative sul lato server.

Cold Fusion, tecnologia per il lato server delle applicazioni web, alternativa a CGI ed ASP

Cookie, pacchetto di informazioni, residente sul client o sul server, che contiene informazioni su alcuni parametri della sessione di comunicazione tra client e server.

CORBA, modello architetturale per applicazioni object oriented in ambiente distribuito.

CSS, Cascade Style Sheet, linguaggio compreso in HTML che permette di definire degli stili comuni a più pagine di un'applicazione web. Uno stile può comprendere tutta una serie di informazioni sulla visualizzazione di una pagina web.

DOS, Disk Operating System, sistema operativo prodotto dalla Microsoft, sostituito poi da Windows, adatto per ambienti monoutente e monoprocesso.

DSSSL, Document Style Semantics and Specification Language, standard, compreso in SGML, al quale si rifanno i linguaggi che definiscono i fogli di stile, come XSL

DTD, Document Type Definition, componente di un'applicazione web che utilizza il linguaggio ipertestuale XML, nel quale sono contenuto le informazioni di struttura dei dati.

FTP, File Transfer Protocol, protocollo di trasferimento dati basato su file. Si posiziona al livello più alto della pila ISO / OSI, così come HTTP

HTML, HyperTextual Markup Language, linguaggio che permette la definizione di un documento ipertestuale, così come XML. Rappresenta il substrato di molte delle più diffuse tecnologie attualmente in uso per le applicazioni web, come ASP, CGI, PHP, ColdFusion.

HTTP, HyperText Transfer Protocol, protocollo di trasferimento dati basato su documenti ipertestuali. Si posiziona, così come FTP, al livello più alto della pila ISO / OSI.

IDL, Linguaggio associato allo standard CORBA, necessario ad interfacciare differenti linguaggi di programmazione.

Internet Explorer, browser prodotto dalla Microsoft, al momento attuale il più utilizzato al mondo.

IIS, Internet Information Server, uno dei Web Server prodotto dalla Microsoft. A differenza di molti degli altri web server, supporta anche la tecnologia ASP.

Java, linguaggio prodotto dalla Sun. Si può considerare come un'estensione del linguaggio C++ particolarmente adatta allo sviluppo in ambiente distribuito.

Javascript, linguaggio di script prodotto dalla Sun, utilizzabile nel lato client di applicazioni web. Si tratta di un linguaggio interpretato, il cui interprete, é spesso contenuto nel browser.

Java Virtual Machine, macchina virtuale in grado di interpretare bytecode Java, come quello che costituisce le applet.

JScript, linguaggio di script prodotto dalla Microsoft, utilizzabile nel alto server di web application. E'spesso utilizzato nell'ambito della tecnologia ASP.

Linguaggio di scripting, linguaggio interpretato. Molti linguaggi di scripting sono a supporto delle applicazioni web: ad esempio javascript, VBScript, JScript, etc.

Middleware, piattaforma software che consente l'interfacciamento in un ambiente distribuito.

Netscape Navigator, browser prodotto dalla Netscape, da sempre antagonista di Microsoft Internet Explorer, ma con il quale condivide la maggior parte delle funzionalità e potenzialità.

ORB, Object Request Broker. E' la piattaforma software che consente in CORBA di implementare I meccanismi di comunicazione tra gli oggetti.

PERL, Processing Extraction Report Language, linguaggio di scripting, utilizzato tra l'altro dalla tecnologia CGI, nel lato server di un'applicazione web.

Pila ISO / OSI,

PHP, Hypertext PreProcessor, tecnologia client-server per la realizzazione di web applications, simile ad ASP, ma non proveniente dalla Microsoft.

RMI, Protocollo applicativo che consente la comunicazione tra oggetti Java remoti.

Servlet, spezzone di un'applicazione Java in esecuzione sul server. Si contrappone all'applet che gira sul client e con essa può collaborare nell'esecuzione di una web application.

SGML, Standard Generalized Markup Language, linguaggio ipertestuale standardizzato. I linguaggi ipertestuali dovrebbero essere sempre sue implementazioni, così come accade per XML.

SQL, Standard Query Language, linguaggio utilizzato per l'interazione con qualsiasi database relazionale.

Tag, componente basilare di un linguaggio ipertestuale. Si riconosce spesso per essere contenuto tra i simboli < e >.

Unix, sistema operativo, alternativo a Windows, freeware. Si caratterizza da sempre per il fatto di essere orientato alle reti di calcolatori.

VBScript, linguaggio di scripting, utilizzato nel lato server di una web application. Si tratta del linguaggio predefinito della tecnologia ASP e si ispira, nella sintassi, a Visual Basic.

Visual Basic, linguaggio della Microsoft che cerca di abbinare la semplicità all'interfacciabilità con la maggior parte dei componenti Microsoft.

Web Server, software che permette di assumere sulla propria macchina il lato server di un'applicazione web.

Windows, sistema operativo prodotto dalla Microsoft. E' un evoluzione del sistema operativo DOS e funge come unico sistema di riferimento per tutte le applicazioni Microsoft.

XML, eXstensible Markup Language, linguaggio ipertestuale moderno, ottenuto dallo standard SGML. La caratteristica che lo differenzia in modo particolare da HTML é l'estensibilità.

XSL, eXtensible Style Language, linguaggio per la definizione dei fogli di stile nell'ambito di XML. Può considerarsi in qualche modo l'equivalente di CSS in ambiente XML.

Bibliografia

[1] F. Pace, *Un tool di reverse engineering per la comprensione di applicazioni web*, Tesi di Laurea in Ingegneria Informatica, Università degli Studi di Napoli "Federico II", marzo 2001

[2] Jim Conallen, *Building Web Application with UML*, Addison Wesley

[3] P. Fraternali, *Tools and approaches for developing data-intensive web applications:A survey,*ACM computing surveys, vol.31, n°3, Settembre 1999

[4] Gnaho C and Larcher F., *A user cebtered methodology for complex and customisable web applications engineering*, http://fistserv.macarthur.uws.edu.au/san/icse99-webe/

[5] Murugesan, S., Deshpande, Y., Hansen S. and Ginige, A. (1999) *Web Engineering: A New Discipline for Development of Web-based Systems*, Proceedings of the First ICSE Workshop on Web Engineering, International Conference on Software Engineering, Los Angeles, Maggio 1999. http://fistserv.macarthur.uws.edu.au/san/icse99-webe/

[6] Murugesan, S., Deshpande, Y., Hansen S (1999) *Web Engineering: Beyond CS, IS and SE*, Proceedings of the First ICSE Workshop on Web Engineering, International Conference on Software Engineering, Los Angeles, Maggio 1999. http://fistserv.macarthur.uws.edu.au/san/icse99-webe/

[7] Ginige A., *Methodologies for developing large and maintainable web based information systems*, http://fistserv.macarthur.uws.edu.au/san/icse99-webe/

[8] Jim Conallen, *Modeling web application architectures with UML*, Communications of the ACM October 1999, vol. 42, n°10

[9] Eun Sook Cho, Soo Dong Kim, Sung Yul Rhew, Sang Duck Lee, Chang Gap Kim, *Object-Oriented web application architectures and development strategies*, 1997 IEEE

[10] Guy M. Nicoletti, *Redefining the Web: Toward the Creation of Large-Scale Distribuited Applications*, 1999 IEEE

[11] Kurt Wallnau, Nelson Weiderman, Linda Northrop, *Distributed Object Technology With CORBA and Java: Key Concepts and Implications*, SEI Giugno 1997

[12] Molly Hammar Cloyd, *Designing User-Centered Web Applications in Web Time*, IEEE Software Gennaio 2001

[13] Shirley A. Becker, Florence E. Mottay, *A Global Perspective on Web Site Usability*, IEEE Software Gennaio 2001

[14] Jonathan Hodgson,*Do HTML Tags Flag Semantic content?*, IEEE Internet Computing Gennaio/Febbraio 2001

[15] Filippo Ricca, Paolo Tonella, *Web site Analysis: Structure and Evolution,*IEEE 2000

[16] Georg Sander, *VCG Visualization of Compiler Graphs User Documentation*, Università di Saarlandes, Germania

[17] Stefano Ceri, Pietro Fraternali, Aldo Bongio, *Web modeling Language (WebML): a modeling language for designing Web sites*, Dipartimento di Elettronica e Informazione, Politecnico di milano, www.polimi.it

[18] Bruce Eckel, *Programmare in C++*,

[19] R. Pressman, *Principi di Ingegneria del software (sec. Edizione),*McGraw-Hill

[20] Ceri, Basi di dati

[21] *OMG Unified Modeling Language Specification*, www.rational.com/uml

[22] Lucidi del corso di Ingegneria del Software , Prof. G.A.Di Lucca, Università degli studi di Napoli Federico II

[23] PDL Guida di Riferimento, http://www.cfg.com/pdl81/

[24] A. Cimitile, U. De Carlini, *Metodologie tecniche e strumenti di reverse engineering*, F. Angeli

[25] JavaScript reference, http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm

[26] HTML 4.0 Specification, http://www.w3.org/TR/1998/REC-html40-19980424/

[27] Built-in ASP Objects Reference, http://msdn.microsoft.com/library/default.asp

[28] MSDN Library per Visual Studio 6

[29] Rigi User's Manual versione 5.4.4, Kenny Wong, http://www.rigi.csc.uvic.ca

[30] A.S. Tanenbaum, *Reti di computer*, Jackson libri

Ringraziamenti

Non si possono qui certamente elencare tutte le persone che hanno fornito un proprio contributo, attivo o passivo, morale o materiale, a questa tesi. Ciò nonostante non ci si può esimere dal ringraziare il relatore Prof..Ing. Di Lucca per la disponibilità e la continua assistenza e il relatore, Prof. Ing. De Carlini per aver reso possibile l'utilizzo di una postazione di lavoro efficiente e all'avanguardia.

Si ringraziano inoltre Fabio Pace per il continuo sprone al lavoro, talvolta esagerato, Roberto Santacroce per aver reso possibile l'utilizzo come caso di studio dell'applicazione web Giuridea, Massimiliano Albanese, Laura Celentano e Maurizio Ferrara, autori del sito web del DIS, utilizzato anch'esso come caso di studio.

Infine non si può non citare tutto il personale del CDS, e in particolare i ragazzi che lavorano allo sviluppo web, per la disponibilità e collaborazione.

Ultimi, ma non per ultimi, vengono tutti i miei colleghi, la mia famiglia e molti dei miei amici, che in questi mesi hanno dovuto conoscere, loro malgrado, i contenuti di questa tesi.