

UNIVERSITÀ DEGLI STUDI DI CATANIA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

CORSO DI LAUREA IN INFORMATICA

GABRIELLA VALERIA CECILIA SANFILIPPO

***UML-based Web Engineering di applicazioni Web per
divisioni della STMicroelectronics***

TESI DI LAUREA

Relatore:
Chiar.ma Prof.ssa Elvinia RICCOBENE

Correlatori:
Dott.ssa Giuliana GANGEMI
Dott. Massimo Orazio SPATA

Alla mia Famiglia

INDICE

INTRODUZIONE	1
CAPITOLO 1	
UML-BASED WEB ENGINEERING.....	4
1. INTRODUZIONE.....	4
2. LA METODOLOGIA DELL'UWE	5
3. DIAGRAMMA DEI CASI D'USO.....	7
3.1 Elementi di modellazione.....	7
3.2 Metodo	7
4. DIAGRAMMA CONCETTUALE.....	8
4.1 Elementi di modellazione.....	8
4.2 Metodo	9
5. MODELLO DELL'UTENTE.....	10
5.1 Elementi di modellazione.....	10
5.2 Metodo	10
5.4 Diagramma dello spazio di navigazione.....	11
5.4.1 Elementi di modellazione.....	12
5.4.2 Metodo.....	13
5.5 Diagramma della struttura di navigazione	14
5.5.1 Inserimento delle primitive d'accesso	14
5.5.1.1 Elementi di modellazione	14
5.5.1.2 Metodo.....	17
5.5.2 Inserimento dei menu.....	18
5.5.2.1 Elementi di modellazione	18
5.5.2.2 Metodo.....	19
6. DIAGRAMMA DI PRESENTAZIONE	21
6.1 Diagramma dell'interfaccia utente astratta	22
6.1.1 Elementi di modellazione.....	22
6.1.2 Metodo.....	24
6.2 Diagramma della Struttura di Presentazione	25
6.2.1 Elementi di Modellazione	26
6.2.2 Metodo.....	27
6.2.2.1 StoryBoarding	28
6.2.2.2 Presentazione Menu_based.....	30
6.2.2.3 Presentazione Map_based.....	31
6.3 Diagramma del Flusso di Presentazione.....	32
6.3.1 Elementi di Modellazione	33
6.3.2 Metodo.....	33
6.4 Diagramma del Ciclo di Vita degli Oggetti.....	34
6.4.1 Elementi di Modellazione	34

6.4.2	Metodo.....	34
7	DIAGRAMMI DI STATO E D'INTERAZIONE PER LA MODELLAZIONE DEGLI SCENARI WEB.....	35
8	DIAGRAMMI D'ATTIVITÀ PER LA MODELLAZIONE DEI TASK.....	36
9	DIAGRAMMA DEI COMPONENTI E DI DISTRIBUZIONE.....	37
10	COSTRUIRE APPLICAZIONI WEB ADATTABILI.....	37
CAPITOLO 2		
IL METAMODELLO UWE.....		39
1	INTRODUZIONE AL METAMODELLO.....	39
2	VINCOLI OCL PER IL METAMODELLO.....	42
2.1	Notazioni.....	42
2.2	Vincoli.....	43
CAPITOLO 3		
ARGOUWE – CASE TOOL PER LA MODELLAZIONE DI APPLICAZIONI WEB.....		50
1	INTRODUZIONE AD ARGOUWE.....	50
2	OPENUWE – TOOL SUITE PER APPLICAZIONI WEB.....	51
3	ARGO UML.....	52
3.1	Perché ArgoUML è diverso.....	54
4	AUTOMAZIONE DEL PROCESSO DI MODELLAZIONE TRAMITE ARGOUWE.....	58
4.1	Verifica della Consistenza.....	61
4.2	Esportazione dei modelli nello standard XMI.....	62
5	COSA NON È STATO IMPLEMENTATO IN ARGOUWE.....	62
CAPITOLO 4		
AMBIENTE DI SVILUPPO E APPLICAZIONI INTEGRATE.....		64
1	STRUTTURA DELL' AMBIENTE DI SVILUPPO.....	64
1.1	Architettura Hardware.....	66
1.2	Architettura Software del Web Server.....	67
1.3	Linguaggi usati.....	68
2	APPLICAZIONI SOFTWARE.....	70
2.1	DivWeb.....	70
2.2	Enterprise Directory.....	72
2.2.1	Cos'è un Directory.....	73
2.2.2	L'organizzazione dei dati in un Directory LDAP.....	73
2.2.3	Directory Distribuite.....	75
2.2.4	LDAP: Il protocollo.....	76
2.2.5	La sicurezza in LDAP.....	78
2.2.6	A cosa serve LDAP?.....	79
2.2.7	Enterprise Directory- Identificazione, Autenticazione, Autorizzazione.....	79

2.3	The Group Service.....	82
-----	------------------------	----

CAPITOLO 5

IL CASO DI STUDIO IP EXCHANGE..... 84

1	INTRODUZIONE.....	84
2	REQUISITI FUNZIONALI.....	84
2.1	Procedure per lo scambio degli IP da automatizzare.....	86
2.1.1	Procedura IP Request.....	86
2.1.2	Procedura IP proposal.....	86
2.1.3	Procedura IP acceptance.....	87
3	REQUISITI NON FUNZIONALI.....	87
4	FUNZIONALITÀ CHE IP EXCHANGE OFFRE AGLI UTENTI.....	87
5	CASI D'USO.....	88
5.1	Attori.....	88
5.2	Casi d'uso in dettaglio.....	90
6	ANALISI E DESIGN DELL'APPLICAZIONE "IP EXCHANGE".....	117
6.1	Diagramma delle classi "type".....	117
6.2	Diagramma Concettuale.....	119
6.3	Diagramma di Navigazione.....	125
6.3.1	Diagramma dello Spazio di Navigazione.....	125
6.3.2	Diagramma della Struttura di Navigazione.....	128
6.4	Diagramma di Presentazione.....	131
7	PROGETTAZIONE DELLA STRUTTURA DELL'APPLICAZIONE.....	137
7.1	Progettazione dettagliata delle classi.....	138
7.2	Descrizione delle classi rilevanti per l'implementazione....	139
7.3	Definizione di sottosistemi e loro interfacce.....	142
8	IMPLEMENTAZIONE.....	143
8.1	Diagramma dei Componenti.....	143
8.2	Diagramma di Dispiegamento.....	151

CAPITOLO 6

IL CASO DI STUDIO DVD-STB WEB..... 152

1	REQUISITI FUNZIONALI.....	152
2	REQUISITI NON FUNZIONALI.....	154
3	FUNZIONALITÀ CHE DVD_WEB OFFRE AGLI UTENTE.....	154
4	CASI D'USO.....	155
4.1	Attori.....	156
4.2	Casi d'uso in dettaglio.....	158
5	ANALISI E DESIGN DELL'APPLICAZIONE "DVD-STB WEB".....	182
5.1	Diagramma Concettuale.....	182
5.1.1	Package DVD.....	182
5.1.2	Package EDA.....	187
5.1.3	Package Analog.....	188
5.1.4	Package Timing Analsis.....	190

5.1.5	Package Training.....	190
5.1.6	Package Notice Board.....	194
5.2	Diagramma di navigazione.....	195
5.3	Diagramma di Presentazione.....	214
6	PROGETTAZIONE DELLA STRUTTURA DELL' APPLICAZIONE.....	222
6.1	Progettazione dettagliata delle classi	222
6.2	Definizione di sottosistemi e loro interfacce	223
7	IMPLEMENTAZIONE.....	223
7.1	Diagramma dei Componenti	224
7.2	Diagramma di Dispiegamento.....	229
	CONCLUSIONI	231
	APPENDICE A	
	ESTENSIONE UML PER APPLICAZIONI WEB.....	233
1	STEREOTIPI PER MODELLARE APPLICAZIONI MULTIMEDIALI GENERICHE	233
2	STEREOTIPI PER MODELLARE CARATTERISTICHE ADATTATIVE	236
	APPENDICE B	
	CORRISPONDENZA FRA METACLASSI SPECIFICHE DI UWE E UML STANDARD	238
	BIBLIOGRAFIA	240

RINGRAZIAMENTI

Alla fine di questo cammino universitario mi sembra doveroso ringraziare tutti i professori per il loro contributo alla mia crescita personale e professionale, in particolare, ringrazio la professoressa Elvinia Riccobene per la disponibilità mostrata in questi mesi.

Ringrazio la divisione DVD della STMicroelectronics, e specialmente gli ingegneri Jim Nicholas e Saeid Azmoodeh, per avermi offerto la possibilità di effettuare questo lavoro. Ulteriori ringraziamenti vanno alle dottoresse Giuliana Gangemi e Rosamaria Balestri ed al dottor Massimo Spata.

Ringraziamenti speciali vanno ai miei genitori per avermi permesso di arrivare alla meta, alla mia famiglia ed agli amici che hanno creduto in me e mi sono stati vicini nei momenti difficili.

INTRODUZIONE

Il World Wide Web (WWW o Web) ha cambiato il modo di vivere e lavorare della società attuale, esso rappresenta lo strumento principale per lo scambio d'informazioni e la condivisione delle conoscenze.

Giorno dopo giorno vengono create delle applicazioni Web sempre più complesse ed importanti. Nonostante ciò, la modellazione d'applicazioni Web è ancora una disciplina nascente, ed i metodi esistenti, specialmente UML (Unified Modeling Language), non supportano la loro modellazione in modo sufficiente.

L'UML-based Web Engineering (UWE) fornisce un profilo dell'UML standard per la modellazione d'applicazioni Web. Quest'approccio d'ingegneria del software è un processo di sviluppo *object-oriented*, incrementale ed iterativo. Supporta l'intero ciclo di vita, dallo studio di fattibilità e progettazione d'applicazioni, all'implementazione e mantenimento (e all'eventuale *project management*).

UWE ambisce ad essere un buon punto di partenza per progettare applicazioni Web, e noi ci proponiamo di verificarlo. Esso possiede le seguenti caratteristiche:

- è *object-oriented*;
- distingue aspetti concettuali, di navigazione e presentazione;
- la sua metodologia indica passo passo le azioni da seguire per la progettazione d'applicazioni Web;
- usa modelli UML;
- propone un'estensione dell'UML basata esclusivamente sull'uso di nuovi stereotipi.

Essendo una metodologia ancora in fase di studio, non esiste un CASE Tool che permetta di sfruttare al massimo tutte le sue potenzialità. Basti pensare che l'unico CASE Tool per il

momento esistente, ArgoUWE [2], non offre molta stabilità e non consente di sfruttare tutte le caratteristiche della metodologia¹.

In questo lavoro di tesi, UWE è stato utilizzato per supportare l'intero processo di sviluppo delle applicazioni Web richieste dal gruppo **CMG** (Consumer Microcontroller Group) della **STMicroelectronics² di Catania**.

A tal fine il lavoro di questa tesi è stato formulato nelle seguenti fasi.

Una **prima fase** di ricerca, necessaria per l'utilizzo di questa metodologia. L'obiettivo è stato acquisire conoscenza e padronanza della metodologia e del CASE Tool relativo, necessarie per poterli applicare ad un caso reale.

In una **seconda fase**, si è effettuata un'analisi dell'ambiente di sviluppo e delle applicazioni **software** già esistenti all'interno dell'azienda (STMicroelectronics), al fine di arrivare ad un'integrazione tra il mio lavoro e i dati raccolti dall'analisi.

La **terza** ed ultima fase è stata dedicata alla modellazione e sviluppo delle applicazioni Web.

Il lavoro svolto in questa tesi, mira a soddisfare, le esigenze delle divisioni **CMGDesign**, **DVD** (Digital Versatile Disk) e **STB** (Set Top Box).

Le necessità di CMGDesign da una parte, e DVD e STB dall'altra sono molto differenti fra loro.

CMGDesign necessita di un'applicazione che supporti l'intero processo (richiesta/consegna) delle procedure esistenti per lo scambio degli IP (Intellectual Property). Scambio che avviene tramite contratti formali fra le divisioni CMG della STMicroelectronics mondiale.

¹ La prima versione di ArgoUWE è stata ufficialmente pubblicata nel settembre del 2003, quando questa tesi era già in fase conclusiva, i diagrammi presentati sono stati aggiornati all'ultima versione del *tool*

² Di seguito contrassegnata come ST.

Il lavoro svolto per le divisioni DVD e STB, mira a rendere visibili le loro attività internamente alle stesse e a consentire scambi d'informazioni e conoscenze fra le divisioni in questione ed il resto della STMicroelectronics mondiale.

Per queste applicazioni è stato seguito l'intero ciclo di sviluppo, cioè la formulazione delle specifiche, studio di fattibilità, creazione del modello ed implementazione.

Questa tesi è composta da sei capitoli strutturati come segue:

Nel **primo capitolo** s'introduce l'approccio proposto dall'UML-based Web Engineering.

Il **secondo capitolo** è dedicato interamente al metamodello UWE.

Nel **terzo capitolo** è presentato ArgoUWE, un CASE Tool per la modellazione d'applicazioni tramite UWE.

Nel **quarto capitolo** vengono descritti l'ambiente di sviluppo e le applicazioni software integrate.

Nel **quinto capitolo** si analizza il caso di studio IP Exchange.

Nel **sesto capitolo** si analizza il caso di studio DVD STB Web.

L'**appendice A** contiene gli stereotipi ed i relativi significati introdotti dall'estensione prodotta da UWE.

L'**appendice B** contiene la corrispondenza fra metaclassi specifiche UWE e UML Standard.

Capitolo 1

UML-BASED WEB ENGINEERING

L'approccio dell'UML-based Web Engineering (UWE) presentato in [21] ed esteso nei seguenti articoli [17] e [28] supporta lo sviluppo d'applicazioni Web, ponendo un'attenzione particolare sulla sistematizzazione, personalizzazione e generazione semi automatica. E' un approccio object-oriented, interattivo ed incrementale basato sull'Unified Modelling Language [44] e sull'Unified Software Development Process [18].

1. Introduzione

Il Web è diventato parte integrante della nostra vita quotidiana. Ogni giorno le applicazioni Web aumentano e diventano sempre più grandi ed importanti. Nonostante ciò la modellazione d'applicazioni Web è ancora una disciplina molto giovane, infatti, i metodi esistenti ed i linguaggi, soprattutto l'UML non supportano la progettazione Web sufficientemente.

L'approccio dell'UML-based Web Engineering (UWE) cerca di risolvere questo problema introducendo un'estensione all'UML, usando i meccanismi d'estensione dell'UML stesso.

L'Unified Modeling Language (UML) è un linguaggio ampiamente usato e standardizzato per definire le specifiche, visualizzare, costruire, e documentare sistemi software [7]. E' uno standard nelle industrie ed è anche il linguaggio più usato per il processo d'ingegneria del software di programmi object oriented. In ogni modo, il suo supporto per la costruzione d'applicazioni Web è considerato insufficiente. Perché non esistono, per esempio, né degli elementi standard del modello per rappresentare *menu* oppure indici definiti, né elementi per rappresentare cammini di navigazione fra differenti siti Web.

Una soluzione possibile è estendere l'UML. Esso, infatti, è un linguaggio definito come auto estensibile, giacché i meccanismi d'estensione sono definiti nell'UML stesso. Attraverso questi meccanismi d'estensione, si possono sviluppare soluzioni specifiche per situazioni specifiche, come sistemi in tempo reale o applicazioni Web.

Queste estensioni sono chiamate profili e possono essere standardizzati dall'Object Management Group (OMG).

Uno di questi profili per le applicazioni Web è l'UML-based Web Engineering (UWE) Sviluppato da Koch, Hennicher e Kraus. Il profilo di UWE è rappresentato in maniera schematica nell'Appendice A.

Nell'UWE, gli elementi del modello (in modo speciale le classi) sono stati estesi (per esempio attraverso degli stereotipi), nuove interfacce sono state definite, e quindi è stato introdotto un approccio per modellare applicazioni Web.

Le principali differenze fra la progettazione di una normale applicazione standalone ed un'applicazione Web sono:

- l'eterogeneità del gruppo dei designers;
- la struttura composta da nodi e *link*;
- la necessità di un controllo della navigazione;
- i contenuti multimediali e quindi la presentazione di questi, per esempio fra diversi browser.

UWE perciò introduce un approccio di modellazione che include tre modelli principali ognuno dei quali è focalizzato su un aspetto centrale delle applicazioni Web: il contenuto, la navigazione e la presentazione.

2. La Metodologia dell'UWE

La metodologia dell'UWE è composta da una notazione ed un metodo.

Come notazione si usa l'UML ed un profilo UML "lightweight" sviluppato in [5], [15] e [24]. Un profilo è un'estensione UML basata sul meccanismo d'estensione definito dall'UML stesso. Il profilo usato in UWE include degli stereotipi, definiti per la modellazione degli aspetti di navigazione e di presentazione di un'applicazione Web, che vengono usati per indicare le proprietà descrittive e restrittive degli elementi di modellazione.

Il metodo fornisce delle linee guida per una costruzione dei diagrammi sistematica e graduale. La costruzione avviene tramite un processo di progettazione interattivo ed incrementale. Le

attività di modellazione sono **analisi dei requisiti, progettazione concettuale, di navigazione e di presentazione**, integrate dalla modellazione dei *task* e dei componenti fisici e dalla visualizzazione degli scenari Web. Il metodo consiglia l'uso di vincoli, scritti in Object Constraint Language (OCL), per garantire la correttezza del modello.

Gli aspetti principali di quest'approccio sono:

- l'uso di una notazione standard;
- una definizione precisa del metodo, tramite una descrizione dettagliata delle linee guida da seguire per la costruzione dei diagrammi;
- la specifica di vincoli, per rendere il modello più preciso.

Il vantaggio di usare UML è che esso è un linguaggio usato da molti.

Il vantaggio dell'uso di un profilo UML, ed in particolare in questo caso di un'estensione con stereotipi restrittivi [6], è che qualunque persona con una conoscenza generale di UML può capire facilmente un modello costruito su queste specializzazioni.

In figura 1.1 è raffigurato il modello rappresentato tramite pacchetti UML collegati tramite dipendenze di tipo *trace* (relazione tra processi).

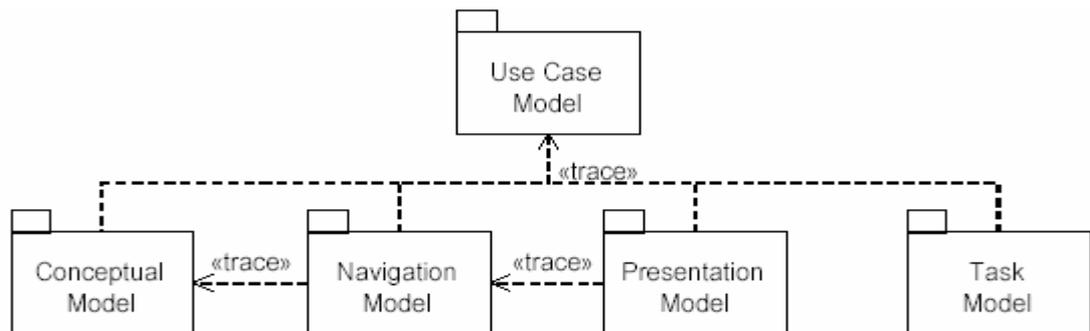


Figura 1.1. Struttura del modello

I principali diagrammi prodotti tramite l'UWE sono:

- Il diagramma dei casi d'uso che cattura i requisiti del sistema.
- Il diagramma concettuale per i contenuti (modello del dominio).

- Il diagramma di navigazione suddiviso in diagramma dello spazio di navigazione e diagramma della struttura di navigazione.
- Il diagramma di presentazione che comprende la modellazione degli aspetti statici e dinamici. Questo diagramma è composto da: diagramma della struttura di presentazione, diagramma del flusso di presentazione, diagramma dell'interfaccia utente astratta, diagramma del ciclo di vita degli oggetti.

Inoltre, la modellazione dei *task* e i diagrammi di stato per gli scenari Web vengono utilizzati per modellare particolari aspetti dinamici dell'applicazione.

3 Diagramma dei casi d'uso

Nell'Unified Software Development Process di Jacobson, Booch e Rumbaugh [18] sono stati proposti i casi d'uso per catturare i requisiti del sistema. Questa è una tecnica, centrata sull'utente, che obbliga a definire gli utenti (attori) dell'applicazione, e propone un modo intuitivo per rappresentare le funzionalità che un'applicazione deve adempiere per ogni attore.

3.1 Elementi di modellazione

I principali elementi di modellazione usati per i diagrammi dei casi d'uso sono: gli *attori* e i *casi d'uso*

Questi possono essere collegati tramite le relazioni d'*ereditarietà*, *inclusione* od *estensione*. Tutti questi elementi di modellazione, insieme ai *pacchetti* ed ai meccanismi per le viste, sono usati in UWE con la stessa semantica e notazione grafica definita in UML.

3.2 Metodo

Per costruire il diagramma dei casi d'uso per un'applicazione Web, si possono seguire i seguenti passi [29], [41]:

1. Determinare gli attori.
2. Per ogni attore si associa un testo adatto alle attività che esso dovrà svolgere.
3. Si raggruppano queste attività in casi d'uso.
4. Si stabiliscono le relazioni fra attori e casi d'uso.
5. Si stabiliscono le relazioni "include" e "extend" fra i casi d'uso.

6. Si semplifica il modello dei casi d'uso definendo le relazioni d'ereditarietà fra attori e/o fra casi d'uso.

Per ogni caso d'uso identificato durante l'analisi dei requisiti è data una descrizione dettagliata attraverso gli scenari (primari e secondari), usando, per esempio, le linee guida di [41]. Il flusso delle attività dei *task* associati ad un caso d'uso può essere rappresentato dai diagrammi d'attività UML.

4 Diagramma Concettuale

La modellazione di applicazioni Web si basa sulla specifica dei requisiti, così come la modellazione di applicazioni software generiche. UWE, essendo un approccio basato sull'UML, propone i casi d'uso per catturare i requisiti.

La progettazione concettuale del dominio è basata su questi casi d'uso e include gli oggetti coinvolti nelle attività che l'utente eseguirà con l'applicazione. Lo scopo di questo diagramma è di costruire un modello del dominio cercando di non prendere in considerazione il cammino di navigazione, la presentazione e gli aspetti d'interazione. Aspetti che saranno analizzati nei rispettivi passi di navigazione e di presentazione della progettazione.

4.1 Elementi di modellazione

Gli elementi principali usati nel modello concettuale sono: *le classi* e *le associazioni*. Queste sono rappresentate graficamente tramite la notazione UML [18]. Se questo modello è formato da molte classi è buona norma raggrupparle usando i *package* della notazione UML.

- *Classi*

Una classe è caratterizzata da un nome, da attributi, operazioni e varianti. Il parametro opzionale varianti viene aggiunto a qualche classe del diagramma concettuale. Questo conterrà delle informazioni aggiuntive usate per la funzionalità di adattamento dei contenuti, per esempio per presentare contenuti diversi o aggiuntivi all'utente in accordo al suo ruolo nell'applicazione. La rappresentazione grafica di una classe con varianti è mostrata in figura 1.2.

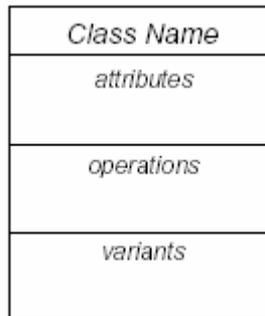


Figura 1.2. Classe con varianti

- **Associazioni e package**

Le associazioni ed i package sono usati come nei diagrammi delle classi UML.

Le classi definite in questo passo saranno usate durante la progettazione della navigazione per derivare i nodi della struttura Web. Le associazioni saranno usate per derivare i link.

4.2 Metodo

Per le applicazioni Web sono estremamente importanti le informazioni che sono scambiate tra l'utente e il sistema. La descrizione dei casi d'uso fornisce un *input* per la progettazione dei contenuti dell'applicazione.

Per ottenere le classi del modello concettuale si eseguono le attività già note della progettazione object-oriented:

1. Trovare le classi.
2. Specificare gli attributi e le applicazioni più importanti.
3. Determinare le associazioni tra le classi.
4. Aggregare le classi e identificare le composizioni di classi.
5. Definire le gerarchie d'ereditarietà.
6. Trovare le dipendenze.
7. Definire i vincoli.

Il risultato di queste attività è il diagramma concettuale del dominio del nostro problema, rappresentato tramite il diagramma delle classi UML.

5 Modello dell'utente

Lo scopo di questo modello è di stabilire quali attributi dell'utente saranno utilizzati per creare un profilo utente, per determinare come questi attributi sono relazionati con altri attributi e come saranno relazionati agli elementi del dominio. Per rappresentare il modello dell'utente dal punto di vista statico, si utilizza un diagramma delle classi. Il diagramma degli stati può essere usato per rappresentare il ciclo di vita degli oggetti degli attributi dell'utente, e per mostrare le dipendenze fra le transizioni degli stati. La maggior parte di queste transizioni sono definite come regole del modello adattativo.

Il modello dell'utente rappresenta la conoscenza, gli scopi e/o le caratteristiche individuali, interessi e compiti di un utente. Questo modello rappresenta la visione che ha il sistema dell'utente. Il suo scopo principale è amministrare i ruoli degli utenti, per utenti o gruppi d'utenti. Come ruolo di un utente s'intende i permessi che l'utente ha all'interno dell'applicazione.

5.1 Elementi di modellazione

Gli elementi di modellazione usati nel modello dell'utente sono: *classi* ed *associazioni*. Se questo modello è composto da molte classi, è raccomandabile raggrupparle usando i *pacchetti* UML.

5.2 Metodo

La costruzione del modello dell'utente è simile a quella del modello del dominio. I requisiti individuati tramite i casi d'uso possono essere utili per individuare le classi del modello dell'utente. I passi principali della metodologia utilizzabile per la costruzione di questo modello sono:

1. Individuare quali attributi, che descrivono le caratteristiche dell'utente, sono rilevanti per l'applicazione.
2. Definire una classe per ogni attributo ben definito.
3. Determinare l'intervallo di valori degli attributi di queste classi.
4. Stabilire quali attributi sono dipendenti dal dominio, e quali rappresentano la conoscenza di *background*, o delle proprietà dell'utente (model cognitive).
5. Identificare le associazioni fra gli attributi utente dipendenti dal dominio e le classi del modello del dominio.

5 Modello dell'utente

Lo scopo di questo modello è di stabilire quali attributi dell'utente saranno utilizzati per creare un profilo utente, per determinare come questi attributi sono relazionati con altri attributi e come saranno relazionati agli elementi del dominio. Per rappresentare il modello dell'utente dal punto di vista statico, si utilizza un diagramma delle classi. Il diagramma degli stati può essere usato per rappresentare il ciclo di vita degli oggetti degli attributi dell'utente, e per mostrare le dipendenze fra le transizioni degli stati. La maggior parte di queste transizioni sono definite come regole del modello adattativo.

Il modello dell'utente rappresenta la conoscenza, gli scopi e/o le caratteristiche individuali, interessi e compiti di un utente. Questo modello rappresenta la visione che ha il sistema dell'utente. Il suo scopo principale è amministrare i ruoli degli utenti, per utenti o gruppi d'utenti. Come ruolo di un utente s'intende i permessi che l'utente ha all'interno dell'applicazione.

5.1 Elementi di modellazione

Gli elementi di modellazione usati nel modello dell'utente sono: *classi* ed *associazioni*. Se questo modello è composto da molte classi, è raccomandabile raggrupparle usando i *pacchetti* UML.

5.2 Metodo

La costruzione del modello dell'utente è simile a quella del modello del dominio. I requisiti individuati tramite i casi d'uso possono essere utili per individuare le classi del modello dell'utente. I passi principali della metodologia utilizzabile per la costruzione di questo modello sono:

1. Individuare quali attributi, che descrivono le caratteristiche dell'utente, sono rilevanti per l'applicazione.
2. Definire una classe per ogni attributo ben definito.
3. Determinare l'intervallo di valori degli attributi di queste classi.
4. Stabilire quali attributi sono dipendenti dal dominio, e quali rappresentano la conoscenza di *background*, o delle proprietà dell'utente (model cognitive).
5. Identificare le associazioni fra gli attributi utente dipendenti dal dominio e le classi del modello del dominio.

6. Definire i vincoli.

7. Raggruppare gli attributi utente in pacchetti di conoscenza dipendenti dal dominio, conoscenza di fondo, e proprietà cognitive.

Il risultato di queste attività è proprio il modello dell'utente, rappresentato come un diagramma delle classi UML.

5.3 Diagramma di Navigazione

Progettare la navigazione è un passo critico nella progettazione di un'applicazione Web. Persino applicazioni semplici con una struttura gerarchica poco profonda, corrono il rischio di diventare complesse, come risultato dell'aggiunta di nuovi link. Infatti, da una parte link aggiuntivi migliorano la navigabilità; dall'altra, però, incrementano il rischio della perdita dell'orientamento. La costruzione del diagramma di navigazione non è utile solo per la documentazione della struttura dell'applicazione, ma permette anche di avere un incremento più strutturato della navigabilità.

Il diagramma di navigazione è composto da altri due diagrammi:

- Il diagramma dello spazio di navigazione il quale indica *quali* oggetti possano essere visitati dalla navigazione attraverso l'applicazione Web.
- Il diagramma della struttura di navigazione il quale indica *come* questi oggetti saranno raggiunti.

Questi diagrammi verranno descritti nei prossimi sottoparagrafi.

5.4 Diagramma dello spazio di navigazione

Durante il processo di costruzione del modello dello spazio di navigazione lo sviluppatore prende delle decisioni cruciali, come quale vista del modello concettuale è necessaria per la navigazione e quali cammini di navigazione sono richiesti per assicurare le funzionalità delle applicazioni. Queste decisioni si basano sul diagramma concettuale, sui diagramma dei casi d'uso e sui requisiti di navigazione che l'applicazione deve soddisfare.

Si propone un insieme di linee guida per la modellazione dello spazio di navigazione. Una descrizione dettagliata delle associazioni, le loro molteplicità ed i nomi dei ruoli, formano la base per una generazione quasi automatica del diagramma della struttura di navigazione.

5.4.1 Elementi di modellazione

Per la costruzione del diagramma dello spazio di navigazione si usano i tre elementi di modellazione: *le classi di navigazione*, *i nodi esterni*, e *le associazioni di navigazione* le quali esprimono navigabilità diretta. Questi sono gli equivalenti delle pagine (nodi) e dei link nella terminologia Web.

- *Classi di navigazione*

Una classe di navigazione modella una classe la cui istanza sarà visitata dall'utente durante la navigazione. Le classi di navigazione avranno lo stesso nome delle classi concettuali corrispondenti.

Per la loro rappresentazione si usa lo stereotipo UML <<navigation class>>. Le classi di navigazione possono contenere attributi derivati. Questi sono derivati dalle classi concettuali che non sono state incluse nel modello di navigazione. La formula per determinare gli attributi derivati può essere data da un'espressione OCL. Un attributo derivato si denota in UML tramite uno slash (/) posto prima del nome.

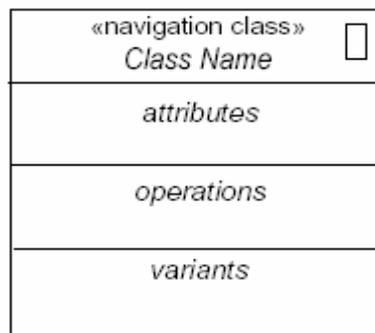


Figura 1.3. Classe di navigazione

- *Nodi esterni*

Un nodo esterno modella un obiettivo di navigazione dipendente da un'altra applicazione Web, per esempio il nodo non fa parte dell'applicazione che si sta modellando, ma può essere raggiunto da un nodo sorgente dell'applicazione in via di sviluppo.

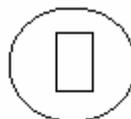


Figura 1.4. Nodo esterno

- ***Navigabilità diretta***

Le associazioni in questo diagramma rappresentano navigabilità diretta dalla classe di navigazione sorgente alla classe obiettivo. La semantica di queste associazioni è diversa da quella delle associazioni del modello concettuale.

Le associazioni in questo modello sono direzionali (o bi-direzionali), per determinare la direzione della navigazione. Queste sono rappresentate da una freccia attaccata in uno o in entrambi capi delle associazioni. Inoltre, ad ogni capo dell'associazione direzionata è associato il nome del ruolo e la relativa molteplicità.

Se nessun ruolo è stato dato all'associazione, si usa la seguente convenzione:

- Se la molteplicità è minore o uguale ad uno, il nome della classe d'arrivo è usato come nome del ruolo;
- Se la molteplicità è maggiore di uno è usato il plurale del nome della classe d'arrivo.

5.4.2 Metodo

Non vi è un metodo standard per l'automazione della costruzione del modello dello spazio di navigazione, esistono diverse linee guida che possono essere seguite dallo sviluppatore, una delle quali è la seguente :

1. Le classi del modello concettuale che sono importanti per la navigazione, sono incluse come classi di navigazione del modello dello spazio di navigazione. Se una classe concettuale non è vista come una classe obiettivo nel diagramma dei casi d'uso, essa è irrilevante nel processo di navigazione e quindi omessa nel diagramma dello spazio di navigazione.
2. Le informazioni, che appartengono alle classi omesse, che sono necessarie per mantenere la funzionalità dell'applicazione, possono essere trattate come attributi delle altre classi nel modello dello spazio di navigazione. Tutti gli altri attributi delle classi di navigazione corrispondono esattamente agli attributi delle classi concettuali corrispondenti. Invece, gli attributi delle classi concettuali che sono considerati irrilevanti per la presentazione sono escluse dal modello dello spazio di navigazione.
3. Le associazioni del modello concettuale sono presenti anche in questo modello. Spesso però, vengono aggiunte delle associazioni per la navigazione diretta in modo da evitare cammini di navigazione con lunghezza maggiore di uno.
4. Si aggiungono delle associazioni basandosi sulla descrizione dei requisiti o sugli scenari descritti dal modello casi d'uso.

5. Si aggiungono dei vincoli per indicare delle restrizioni nello spazio di navigazione.

5.5 Diagramma della struttura di navigazione

Il diagramma della struttura di navigazione indica come la navigazione può essere eseguita usando elementi d'accesso come *indici*, *guided tour*, *query* e *menu*.

Tecnicamente, il cammino di navigazione, insieme con gli elementi d'accesso sono introdotti da un diagramma delle classi che può essere costruito sistematicamente dal modello dello spazio di navigazione in 2 passi. Innanzi tutto, migliorando il modello dello spazio di navigazione introducendo indici, cammini guidati e query. Dopodiché derivando direttamente i menu dal modello ottenuto nel passo precedente. I menu rappresentano le possibili scelte per la navigazione.

Il diagramma della struttura di navigazione risultante definisce la struttura dei nodi e dei link di un'applicazione Web mostrando come la navigazione è supportata dalle primitive d'accesso.

Il seguente paragrafo descrive come costruire il diagramma della struttura di navigazione.

5.5.1 Inserimento delle primitive d'accesso

Le primitive d'accesso indici, *guided tour* e *query* sono dei nodi aggiuntivi necessari per accedere ad istanze delle classi di navigazione. Un'altra primitiva d'accesso sono i menu che saranno trattati nel prossimo paragrafo.

5.5.1.1 Elementi di modellazione

Per descrivere indici, *guided tour* e *query* usiamo la seguente notazione. Gli stereotipi e le icone associate sono state introdotte da Koch & Mandel [23].

- **Indici**

Un indice consente di accedere direttamente ad un'istanza di una classe di navigazione. Esso è modellato attraverso un oggetto composto, il quale contiene un numero arbitrario di index item.

Ogni index item è a sua volta un oggetto con un nome ed un link ad un'istanza di una classe di navigazione. Ogni indice è un membro di una classe indice, la quale è stereotipata da <<index>> ed ha un'icona corrispondente.

Una classe indice deve essere costruita in modo da seguire la struttura composta mostrata in figura 1.5. Lo stereotipo <<index>> è uno stereotipo restrittivo così com'è stato definito da Berner, Glinz e Joos [6].

In pratica, però si usa spesso la notazione breve mostrata in figura 1.6.

Notiamo che nella forma breve l'associazione fra Index e NavigationalClass è derivata dalla composizione degli indici e dall'associazione fra IndexItem e NavigationalClass.

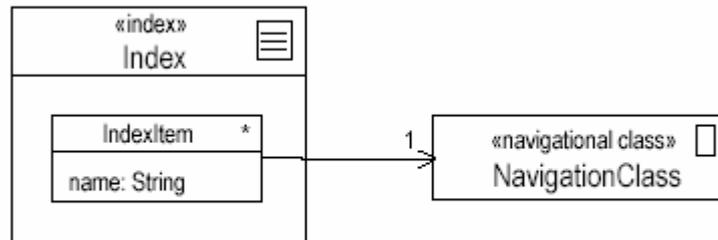


Figura 1.5. Classe indice

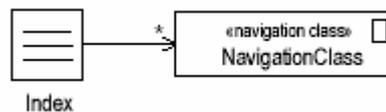


Figura 1.6. Notazione breve per la classe indice

- **Guided tour**

Un *guided tour* consente un accesso sequenziale ad istanze di una classe di navigazione. Per le classi che contengono oggetti *guided tour* si usa lo stereotipo <<guidedTour>> e la sua icona corrispondente è mostrata in figura 1.7. Ogni classe *guided tour* deve essere costruita conformemente alla struttura di composizione delle classi mostrata in figura 1.7, nella quale ogni elemento Next deve essere connesso ad una classe di navigazione. I *guided tour* possono essere controllati dall'utente o dal sistema.

Esiste anche la notazione breve nella quale, ogni classe *guided tour* deve essere connessa ad una classe del modello di navigazione tramite un'associazione diretta che ha la proprietà ordered, questa notazione è mostrata in figura 1.8.

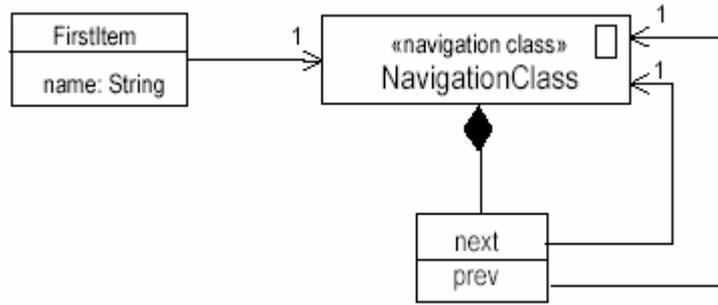


Figura 1.7. Classe guided tour

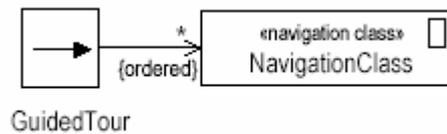


Figura 1.8. Notazione breve per la classe guided tour

- **Query**

Un *query* è modellata da una classe che ha una stringa di *query* come attributo. Questa stringa può essere data, per esempio, da un'operazione di selezione del linguaggio OCL. Per le classi *query* si usa lo stereotipo <<query>> e l'icona corrispondente è quella in figura.

Come mostrato in figura 1.9, ogni classe *query* è la sorgente di due associazioni dirette collegate attraverso il vincolo {xor}. In questo modo possiamo modellare il fatto che un query con molti oggetti come risultati deve prima essere condotta ad un indice che supporta la selezione di una particolare istanza di una classe di navigazione. Il risultato di un *query*, potrebbe essere usato come input per un *guided tour*.

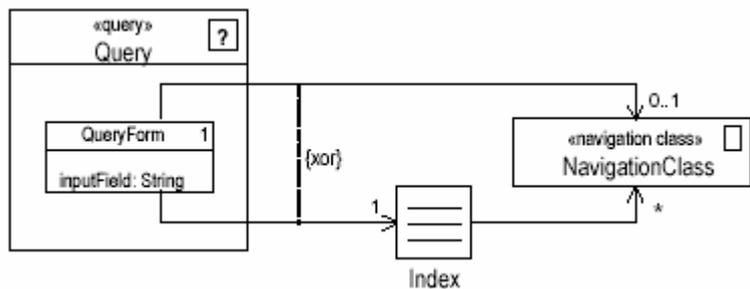


Figura 1.9. Classe query

La figura 1.10 mostra la notazione breve per una classe *query* combinata con un indice o con un *guided tour*:

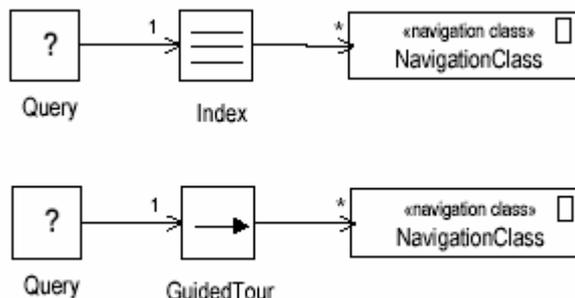


Figura 1.10. Notazione breve per la classe query

5.5.1.2 Metodo

Il raffinamento del modello dello spazio di navigazione tramite l'inserimento delle primitive di accesso di tipo indice, guided tour e query segue certe regole che possono essere sintetizzate come segue:

1. Le associazioni bi-direzionali, che hanno molteplicità maggiore di uno ad entrambi gli estremi, sono sostituite con due associazioni unidirezionali.
2. Le associazioni bi-direzionali, che hanno molteplicità maggiore di uno ad un solo capo, sono sostituite con un'associazione diretta verso l'estremo con molteplicità maggiore di uno. La navigazione nell'altra direzione è garantita dall'uso dell'albero di navigazione, che sarà introdotto dopo nel progetto.
3. Si considerino solo quelle associazioni unidirezionali del modello dello spazio di navigazione che hanno molteplicità maggiore di uno.
4. Per ogni associazione di questo tipo, si scelga una o più primitive di accesso per realizzare la navigazione.
5. Il modello dello spazio di navigazione è quindi completato con le primitive d'accesso corrispondenti. I nomi dei ruoli della navigazione nel modello dello spazio di navigazione adesso sono spostati nelle primitive d'accesso.
6. Si aggiungano dei vincoli per modellare invarianti e condizioni. Questi sono dedotti dalla descrizione dettagliata del modello dei casi d'uso.

Nel passo 4 il compito del progettista è di scegliere una primitiva d'accesso appropriata. Si noti che è anche possibile, per automatizzare completamente questo passo, scegliere un indice di *default* per il progetto come in accordo ad un attributo che abbia la proprietà {key} e che punti ad una classe di navigazione obiettivo.

5.5.2 Inserimento dei menu

In questo passo, le primitive d'accesso di tipo menu sono aggiunte al modello della struttura di navigazione.

5.5.2.1 Elementi di modellazione

I menu sono delle primitive d'accesso addizionali che possono essere aggiunti a quelli che abbiamo introdotto precedentemente. Lo stereotipo UML <<menu>> è stato definito in [23] ed appartiene agli stereotipi restrittivi concordemente alla classificazione di stereotipi data da [6].

- *Menu*

Un menu è un indice o un insieme di elementi omogenei, come un indice, un guided tour, un query, un'istanza di una classe di navigazione o un altro menu. Sono modellati tramite degli oggetti composti che contengono un numero finito di menu item. Ogni menu item ha un nome costante ed un link che può puntare o ad un'istanza di una classe di navigazione o ad una primitiva d'accesso.

Ogni menu è un'istanza di una qualche classe menu, la quale è stereotipata da <<menu>> ed ha un'icona corrispondente.

Una classe menu deve essere costruita seguendo la struttura di composizione della classe mostrata in figura 1.11. La proprietà {frozen} è attaccata ad ogni nome di attributo in una classe di menu item per indicare che un menu item ha un nome fisso. Tuttavia, la stessa classe di menu item può avere diverse istanze poiché ci possono essere menu item con lo stesso nome ma che puntano ad oggetti diversi.

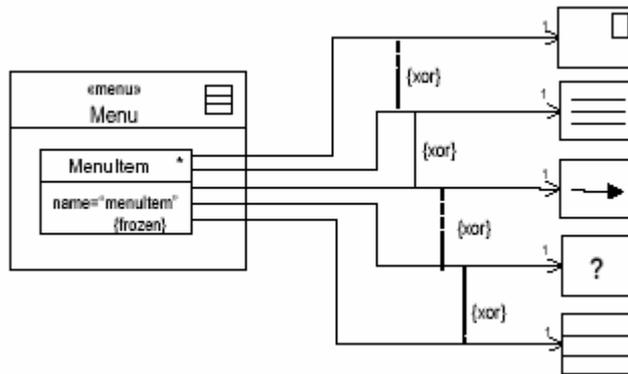


Figura 1.11. Classe menu

Una notazione più pratica della classe menu nel modello dello spazio di navigazione è la notazione breve mostrata in figura 1.12. Questa non segue del tutto il meccanismo di estensione permesso da UML, poiché permette l'utilizzo di un numero variabile di scompartimenti con dentro il nome di un menu item. Per essere una notazione UML compatibile al 100% si utilizza quella mostrata in figura 1.11, questa ha lo svantaggio di consumare molto spazio.

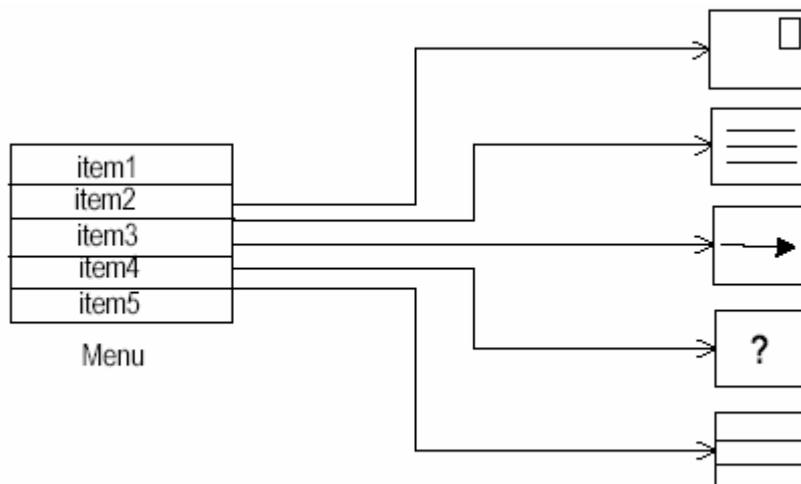


Figura 1.12. Notazione breve per la classe menu

5.5.2.2 Metodo

Le primitive di accesso menu possono essere aggiunte al modello dello spazio di navigazione attraverso le seguenti regole:

1. Si considerino quelle associazioni, che hanno come sorgente una classe di navigazione.
2. Si associ ad ogni classe di navigazione, che ha (nell'ultimo diagramma ottenuto) almeno un'associazione uscente, una classe menu corrispondente. L'associazione fra la classe di navigazione e la sua classe menu è una composizione.
3. Si riorganizzi un menu con degli eventuali sottomenu.
4. Si introduca per ogni ruolo, ai capi delle associazioni dirette introdotte nel modello precedente, un corrispondente menu item. Per default, il nome del ruolo è usato come nome del menu item.
5. Ogni associazione che nel modello precedente ha come sorgente una classe di navigazione adesso diventa un'associazione del menu item corrispondente introdotto nel passo precedente.
6. Si aggiungano dei vincoli per aggiungere precisione al modello.

Tutti i passi del metodo precedentemente esposto possono essere eseguiti in modo automatico. Come risultato si ottiene il modello della struttura di navigazione della nostra applicazione Web. Il metodo garantisce che questo modello è conforme al *pattern* di figura.

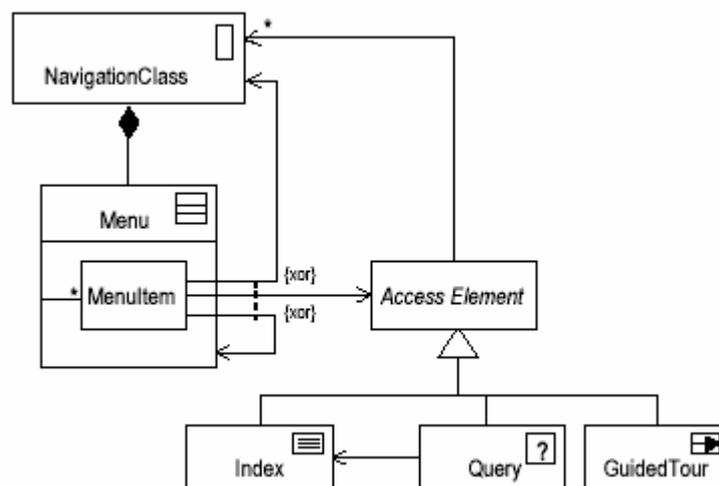


Figura 1.13. Pattern per le strutture d'accesso

6 Diagramma di Presentazione

Il diagramma di presentazione descrive *dove* e *come* gli oggetti di navigazione e le primitive d'accesso saranno presentate all'utente. La progettazione della presentazione supporta la trasformazione del diagramma della struttura di navigazione in un insieme di diagrammi che mostrano come gli oggetti saranno visti dall'utente in modo statico, una rappresentazione schematica di questi oggetti (pagine Web in progettazione) ed il comportamento dinamico di questi. La rappresentazione schematica è simile alla tecnica di schizzo usate da qualche progettista dell'interfaccia utente, che però non ha una notazione precisa per essa così come è stato descritto da [39]. UWE definisce una notazione formale per questi schizzi, usando per la rappresentazione grafica la notazione UML.

Il modello di presentazione punta ad un'organizzazione strutturale della presentazione e non all'apparenza fisica vera e propria, cioè ai caratteri, formati speciali ecc. Queste decisioni saranno prese in fase d'implementazione. Comunque il *layout* degli elementi del modello da un suggerimento circa la posizione e la grandezza degli elementi.

Il sottoparagrafo seguente mostra come il diagramma di presentazione è derivato dal diagramma della struttura di navigazione. Inoltre sono aggiunte delle informazioni acquisite durante l'analisi dei requisiti. Gli aspetti statici sono modellati dal diagramma della struttura di presentazione e dal diagramma dell'interfaccia utente astratta. Questo diagramma mostra come le finestre (o *window*), i *frame* siano riempite dai contenuti e quali contenuti possono essere mostrati simultaneamente. Il diagramma dell'interfaccia astratta mostra uno schizzo del contenuto di ogni nodo.

Questo diagramma può produrre diverse implementazioni, tutto dipende dalla restrizione della piattaforma e dalla tecnologia usata. Si possono generare, quindi pagine statiche e dinamiche, pagine *client* e *server* ad una o più finestre, ecc. Un concetto importante del diagramma di presentazione sono le finestre i *frameset* e *frame*. L'uso dei *frame* permette anche la visualizzazione dello spazio di visualizzazione, di solito presentato come un albero di navigazione (detto anche mappa di navigazione).

Gli aspetti dinamici della presentazione sono modellati usando il diagramma di flusso di presentazione e, in modo opzionale, il diagramma del ciclo di vita degli oggetti. Per

rappresentare il diagramma di flusso di presentazione si usa il diagramma di sequenza UML, mentre per il diagramma del ciclo di vita degli oggetti si usa il diagramma di stato. Entrambi questi diagrammi descrivono il comportamento degli oggetti di presentazione, per esempio la variazione nell'interfaccia utente quando l'utente interagisce con essa o quando il sistema reagisce ad eventi interni come i *timeout*. La costruzione del diagramma del flusso di presentazione è consigliata quando si usa una tecnica a finestra multipla o con frame. Questo specifica quando le finestre sono aperte, chiuse e quando i frame cambiano i loro contenuti. Il diagramma del ciclo di vita degli oggetti descrive il comportamento di oggetti critici e come la loro transizione influenza lo stato di altri oggetti.

6.1 Diagramma dell'interfaccia utente astratta

L'obiettivo del diagramma dell'interfaccia utente astratta è di dare una tecnica e una notazione per lo schizzo dell'interfaccia utente, ad esempio come il contenuto di un nodo (pagina Web) è presentata ad un utente. Il progetto di questa interfaccia astratta, come detto sopra, modella principalmente l'organizzazione strutturale della presentazione, in termini di testo, immagini, form e menu, e non le caratteristiche del layout: font, colori e formati speciali ecc. Queste decisioni saranno prese durante lo sviluppo di un prototipo dell'interfaccia utente o nella fase di implementazione. Questo modello fornisce comunque qualche suggerimento circa la posizione e la dimensione degli oggetti dell'interfaccia fra di loro. Per costruire il modello di presentazione prima di tutto bisogna decidere quali elementi di modellazione saranno usati per la presentazione delle classi di navigazione e quali per la presentazione delle primitive d'accesso.

La progettazione dell'interfaccia utente astratta può essere considerato come un passo opzionale, perché le decisioni circa la progettazione dell'interfaccia utente possono essere prese durante la realizzazione dell'interfaccia stessa. Comunque la produzione di schizzi di questo tipo è utile in una discussione con il cliente circa le caratteristiche del prodotto futuro.

6.1.1 Elementi di modellazione

Le istanze delle classi di presentazione sono dei contenitori, che comprendono elementi di modellazione come testo, immagini, form, bottoni, sequenze video, sequenze audio, ancore, liste di testo, immagini, ecc. Una classe di presentazione segue le regole di composizione

mostrate in figura 1.14 . I seguenti elementi di modellazione (dieci classi stereotipate) sono proposte per descrivere l'interfaccia utente astratta di un'applicazione Web. Gli stereotipi per testo, form, bottoni, immagini, video, ancore, collezioni e collezioni di ancore sono mostrati in figura e sono stati introdotti da Baumeister [24].

- **Classe di presentazione**

Una classe di presentazione modella la presentazione di una classe di navigazione o di una primitiva d'accesso o di una composizione di classi di presentazione. Una classe di presentazione è anche un contenitore per un insieme di altre classi che modellano la presentazione degli attributi di una classe di presentazione. Questa è stereotipata da <<presentation class>> con la sua corrispondente icona mostrata in figura 1.14.

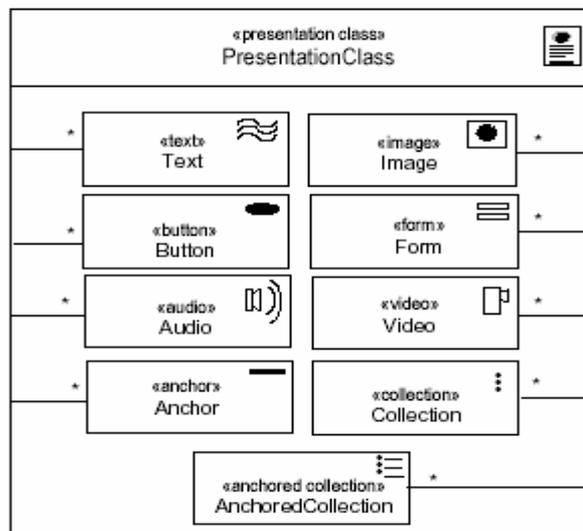


Figura 1.14. Classe di Presentazione

- **Text**

Un testo è una sequenza di caratteri.

- **Ancora**

Un'ancora è un area cliccabile, rappresenta un punto di partenza per una navigazione, quindi ha associato un link ad un altro nodo. Le ancore sono generalmente presentate in letteratura come una parte di un link, raramente come un oggetto indipendente. Un'ancora è formata da una elemento di presentazione e da un link. L'elemento di presentazione può essere un testo, un'immagine, un video, un oggetto interattivo o un intero documento.

- ***Bottone***

Un bottone è un'area cliccabile, che ha un'azione associata. Esempi di azione possono essere visualizzare un'immagine, eseguire un applet, ecc..

- ***Immagini, audio e video***

Immagini, audio e video sono oggetti multimediali. Le immagini possono essere visualizzate; sia gli audio sia i video possono essere avviati, fermati, portati avanti e indietro. Per fornire questa funzionalità, gli oggetti dell'interfaccia utente predisposti per l'interazione, come bottoni o ancore, devono essere associati a questi oggetti multimediali.

- ***Form***

I *form* sono usati per richiedere informazioni all'utente. L'utente inserirà le informazioni in uno o più campi di input, o selezionerà un'opzione ecc.. La semantica di questi elementi del modello include la visualizzazione dei contenuti, l'attesa per un'attività utente, la valutazione degli input e l'esecuzione di un evento definito.

- ***Collections e anchored collections***

Una *collection* è formata da un insieme di elementi testo. *Un'anchored collection* è un insieme d'ancore.

6.1.2 Metodo

Il diagramma dell'interfaccia utente astratta è composto da una serie di diagrammi delle classi, rappresentate principalmente come composizioni. Ogni composizione modella una classe di presentazione usando un template per rappresentare i contenuti. Quindi una classe di presentazione (composizione) o le classi di presentazione che sono inserite in ogni frame di una pagina Web, costituiscono un template per una pagina Web.

Le seguenti regole possono essere usate come linee guida per la costruzione del diagramma dell'interfaccia utente astratta:

1. Si costruisca una classe di presentazione per ogni classe di navigazione presente nel modello della struttura di navigazione. La classe di presentazione definisce un template soddisfacente per presentare le istanze della classe tenendo conto anche dei suoi attributi. Le classi stereotipate, come <<text>>, <<image>>, <<audio>>, <<video>> sono usate per rappresentare gli attributi di tipo primitivo, mentre le <<collections>> sono usate per le liste.

2. Si costruisca una classe di presentazione per ogni indice e menu presenti nel modello della struttura di navigazione. La presentazione di classi menu o indice consiste, di solito, in una lista di ancore. Si usino quindi gli stereotipi <<anchored collection>> e <<anchor>> rispettivamente.
3. Si costruisca una classe di presentazione per ogni *querye guided tour*. Per le *query* si usi lo stereotipo <<form>> per i *guided tour* s'introducano degli elementi di menu aggiuntivi ("next" e "prev") che permettano di navigare verso il prossimo ed il precedente oggetto usando un guided tour.
4. Si aggiungano delle ancore, a quelle classi di presentazione che permettono la creazione, la distruzione o l'esecuzione di operazioni su oggetti del modello concettuale. I requisiti funzionali di queste ancore provengono dal modello dei casi d'uso.
5. Si aggiungano dei vincoli OCL se necessario.

Deve essere assicurato che esista solo un insieme finito di cammini di navigazione dalla classe radice di ogni navigazione o dalla classe indice. A questo scopo si assume che il diagramma della struttura di navigazione non abbia cicli, cioè sia, per esempio un grafo diretto aciclico. Questa non è esattamente una restrizione, ma in ogni caso è prevista una presentazione dell'albero di navigazione, il quale permette eventualmente di muoversi a ritroso.

Per quanto riguarda la presentazione dell'albero di navigazione è ovvio che, in pratica, la profondità dell'albero deve essere limitata. Per una rappresentazione di quest'albero si possono anche usare diversi *frame*

6.2 Diagramma della Struttura di Presentazione

Il diagramma della Struttura di presentazione descrive in modo statico *dove* gli oggetti di navigazione e le primitive d'accesso saranno presentate all'utente. Quindi, lo scopo di questo passo è di specificare se è usata la tecnica a finestra singola o multipla, come sono divisi i vari *frame* e *frameset* (se si è deciso di usare i frame) ed in quale frame o finestra il contenuto è visualizzato.

6.2.1 Elementi di Modellazione

I seguenti elementi di modellazione sono usati per descrivere la struttura di presentazione delle applicazioni Web. *Window* (o finestre), *frameset* e *frame* sono usati per descrivere la posizione della presentazione, mentre le classi di presentazione (definite nella sezione precedente) sono usate per descrivere il contenuto dei nodi.

- *Window*

Una *window* è l'area dell'interfaccia utente nella quale sono visualizzati gli oggetti di presentazione. Una finestra si può spostare, massimizzare/minimizzare, ridimensionare, ridurre ad icona, o chiudere. Essa include al più cinque bottoni, uno per trasformare la finestra ad icona, uno per chiuderla, uno per massimizzarla/minimizzarla, ed uno per ridimensionarla. In oltre una finestra include le *scrollbar* verticali ed orizzontali, le quali permettono la visualizzazione dell'intero contenuto della finestra. Ogni finestra è un'istanza di una classe stereotipata con <<window>> la quale è stata costruita secondo la struttura mostrata in figura 1.15. Una finestra può essere organizzata come una gerarchia.

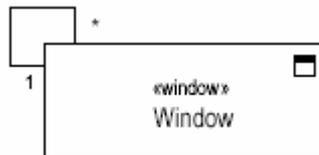


Figura 1.15. Classe window

- *Frameset*

Un *frameset* è un elemento di modellazione usato per definire diverse aree di visualizzazione all'interno di una finestra. Un *frameset* è sempre contenuto all'interno di una finestra; è diviso in elementi di più basso livello - chiamati *frame* - e potrebbe anche contenere un numero arbitrario di *frameset* annidati. Un *frameset* è un'istanza di una classe stereotipata da <<frameset>> con la sua icona corrispondente mostrata in figura 1.16.

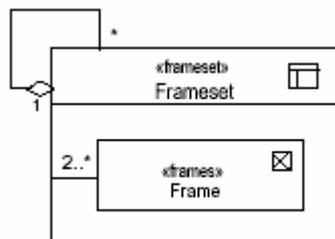


Figura 1.16. Classe Frameset

- **Frame**

Un *frameset* è diviso in un una serie di *frame*. Un *frame* è un istanza di una classe *frame* stereotipata da <<frame>> con la sua corrispondente icona mostrata nella figura 1.16. *Frameset* e *frame* devono essere costruiti in accordo alla struttura mostrata nella figura 1.16.

- **Present**

Indica che l'oggetto d'arrivo dell'associazione è visualizzato nella locazione indicata dall'oggetto sorgente.

Un diagramma di presentazione per un'applicazione Web è costruito attraverso le classi stereotipate <<window>>, <<frameset>>, <<frame>> e <<presentation class>>, sono anche incluse le associazioni di tipo <<present>> in accordo con il *pattern* mostrato in figura 1.17. Secondo il *pattern* in figura si può definire un solo *frameset* per finestra.

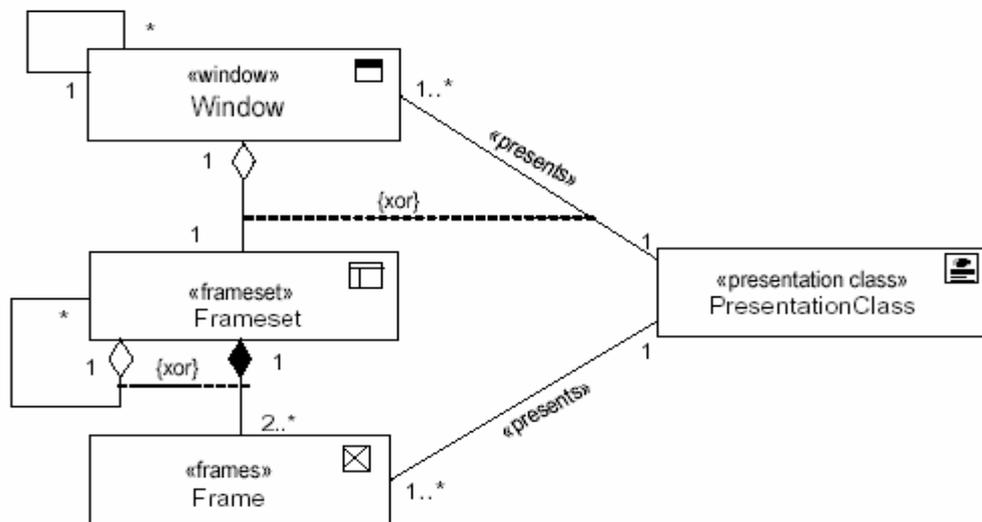


Figura 1.17. Pattern per gli elementi di presentazione

6.2.2 Metodo

Per costruire il diagramma della struttura di presentazione, il progettista deve prendere delle decisioni quali il numero di finestre da usare, il numero di *frame* in cui un *frameset* è diviso. La costruzione del modello della struttura di presentazione non può essere automatizzato completamente, ci sono comunque delle linee guida che il progettista può seguire:

1. Si scelga la tecnica da usare: a finestra singola o multipla. Nel caso in cui si scelga la tecnica a finestra multipla si pianifichi quante finestre si dovranno usare.

2. Si scelga lo stile del *frame*, cioè con o senza *frameset*. Nel primo caso si specifichi quanti *frame* ha ogni *frameset*.
3. Si usino le classi di presentazione costruite per le primitive d'accesso e per ogni classe di navigazione nel diagramma precedente.
4. Si decida in quale *frame* o *frameset* o finestra (nello stile senza *frame*) ogni classe di presentazione è presentata all'utente.
5. Si usi l'associazione <<presents>> per le relazioni create nel passo 4 fra finestre o *frame*, e classi di presentazione.
6. Si costruisca un diagramma delle classi usando la composizione per mostrare quale sarà il layout della pagina Web.

Se si usano molte finestre e/o *frame*, è opportuno costruire delle viste parziali del modello della struttura di presentazione, per evitare di sovraccaricarlo.

Con questo metodo possono essere modellati diversi tipi di presentazione, ad esempio tre tipi diversi di presentazione sono: *storyboarding menu based, map based*

6.2.2.1 StoryBoarding

Il progetto di uno *storyboarding* può essere considerato un passo opzionale. Il punto d'inizio per questo tipo di modellazione è lo schema delle varie interfacce utente (una per ogni classe di presentazione). Quando sono state prodotte tutte le viste per le interfacce, gli scenari di *storyboarding* possono essere sviluppati mostrando le sequenze di viste di interfacce utente nell'ordine in cui saranno raggiunte durante la navigazione [36].

Sia gli schizzi delle interfacce utente che gli scenari di *storyboarding* sono utili per la comunicazione fra il cliente ed il progettista Web.

Le interfacce sono modellate in modo simile a come visto per il diagramma dell'interfaccia utente astratta, tramite classi ed associazioni di composizione UML. Per la visualizzazione si sceglie la composizione nidificata (figura 1.14) come alternativa offerta da UML alla composizione disegnata attraverso il diamante nero.

Elementi di modellazione

Per la costruzione di schizzi sono proposti una serie di elementi di modellazione, alcuni dei quali sono mostrati in figura 1.18.

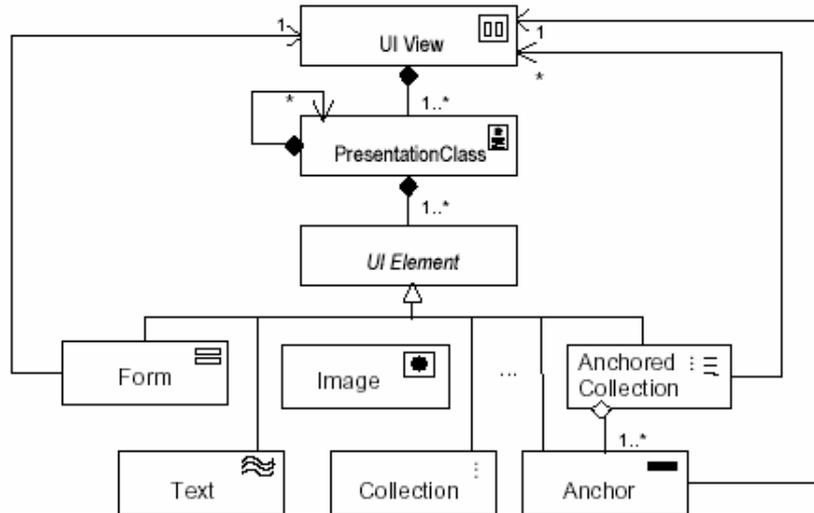


Figura 1.18. Metamodello per le gli elementi delle interfacce utente astratte

- **User Interface View**

Una classe *user interface view* indica che, ogni istanza di questa classe è un contenitore di tutti gli elementi astratti dell'interfaccia che sono mostrati contemporaneamente all'utente. Per le classi della vista dell'interfaccia si usa lo stereotipo <<UI view>> come mostrato in figura 1.18.

- **Presentational class**

Una *presentational class* o classe di presentazione, è un'unità strutturale che permette di dividere una vista dell'interfaccia utente in più *user interface element*. Per la classe di presentazione vale ancora la definizione data sopra in figura 1.14 equivalente a quella data in figura 1.18.

- **User interface elements**

Una classe *user interface element* è una classe astratta che ha diverse specializzazioni che descrivono i particolari elementi d'interfaccia (figura 1.18). Questi elementi sono già stati definiti nella figura 1.14.

Metodo

Per la creazione di uno *storyboarding* si devono aggiungere i seguenti passi a quelli relativi al metodo per costruire il diagramma dell'interfaccia utente astratta:

6. Si determinino quali elementi di presentazione devono essere presentati in una finestra. Le classi di presentazione corrispondenti devono essere composte in un *user interface view* (<<UI view>>). Di solito l'utente necessita di una combinazione di dati concettuali ed elementi che rendono la navigazione più facile, tipicamente un *user interface view* è composto di una classe di presentazione costruita per una classe di navigazione, ed una classe di presentazione per gli altri elementi.
7. Si costruisca uno scenario di *storyboarding* rappresentati da sequenze di <<UI view>>. A questo scopo si aggiungono dei link che connettono un'ancora di un UI view con un altro UI view, in modo da mostrare i possibili flussi di presentazione che possono essere provocati dall'interazione dell'utente.

6.2.2.2 *Presentazione Menu_based*

Con questa tecnica si usano i *frameset* per partizionare la presentazione in *frame*. Il *frame* di sinistra rappresenterà, sempre, il menu principale; mentre il frame di destra mostrerà il contenuto desiderato.

Metodo

Per la creazione di una presentazione *menu_based* si devono aggiungere i seguenti passi a quelli relativi al metodo per costruire il diagramma dell'interfaccia utente astratta:

6. Si scelga una classe di navigazione come radice per la navigazione, e si costruisca la classe di presentazione per il menu relativo a questa classe di presentazione (menu principale).
7. Si unisca la classe di presentazione (costruita al passo 1) con la classe di presentazione del suo menu (costruita nel passo 6) in un frameset.

6.2.2.3 *Presentazione Map_based*

Nella presentazione *map_based*, sono usati, ancora, i *frameset* i quali permettono la visualizzazione della struttura di navigazione.

L'idea è sempre quella di dividere la presentazione in due parti principali: una che fornisce la presentazione dell'albero di navigazione (mostrando il cammino di navigazione dell'utente e quindi il contesto di navigazione), e l'altra che mostra, invece, il contenuto desiderato.

Metodo

Su queste basi, si definisce una procedura per derivare il modello *map_based*. Oltre ai soliti passi relativi al metodo per costruire il diagramma dell'interfaccia utente astratta, si considerino i seguenti:

1. Si scelga una classe di navigazione come radice.
2. Per ogni classe di navigazione e per ogni indice si considerino tutti i possibili cammini (del diagramma della struttura di navigazione) dalla classe radice alla classe in questione. Per ogni cammino si costruisca una presentazione dell'albero di navigazione corrispondente.
3. Si combinino i risultati dei passi relativi al metodo per costruire il diagramma dell'interfaccia utente astratta ed il passo 2 in *frameset*. Ogni *frameset* è diviso in due parti: il frame di destra contiene la presentazione della classe di navigazione o della classe indice costruita nei primi passi; il frame di sinistra rappresenta l'albero di navigazione (costruito nel passo 2) corrispondente ad un possibile cammino di navigazione verso questa classe.

Come detto precedentemente, deve essere assicurato che esista solo un insieme finito di cammini di navigazione dalla classe radice d'ogni navigazione, o dalla classe indice. A questo scopo si assume che il diagramma della struttura di navigazione non abbia cicli, cioè sia per esempio un grafo diretto aciclico.

6.3 Diagramma del Flusso di Presentazione

Il diagramma del flusso di presentazione descrive in quale finestra, *frameset*, o *frame* gli oggetti di presentazione sono posti e come il flusso dei controlli passa da un elemento all'altro. In questo diagramma sono rappresentati aspetti di navigazione e presentazione in modo combinato.

In un particolare istante un oggetto di presentazione è attivo se esso è incluso in una finestra o *frame* attiva. Una finestra è attiva se il *mouse* punta su questa finestra (o *frame*). Solo una finestra o *frame* è attiva in un determinato istante. Un oggetto di presentazione è percepibile se esso è incluso in una finestra in un certo periodo. Per percepibile s'intende che nel caso in cui l'oggetto sia un file audio questo è ascoltabile, oppure se si tratta di un video è visibile, ecc.

Se si usa la tecnica a finestra singola, l'unico *frameset* ad essere attivo è quello incluso nell'unica finestra esistente, e quindi tutti gli oggetti di presentazione inclusi in questo *frameset* (o nei *frame* del *frameset*) sono percepibili (o visibili). La transizione ad un altro oggetto di presentazione implica la rimozione dell'oggetto correntemente in uso per visualizzare quello nuovo.

Se si usa la tecnica a finestra multipla si possono verificare le seguenti situazioni:

- Un oggetto di presentazione che si trova nella finestra attiva, è rimpiazzato dal nuovo oggetto, quindi la stessa finestra rimane attiva.
- Un oggetto di presentazione è visualizzato in una nuova finestra aperta proprio per contenere questo oggetto nuovo, il contenuto dell'altra finestra rimane invariato, ma la nuova finestra è adesso quella attiva.
- Una finestra è chiusa senza alterare il contenuto delle altre finestre. Una delle rimanenti finestre sarà segnata come attiva.
- Se le finestre sono dipendenti fra di loro, se una finestra è chiusa allora questo può significare che anche le altre saranno chiuse. Le dipendenze sono basate, usualmente, sulla gerarchia fra finestre.
- Il puntatore del mouse può spostarsi da una finestra all'altra, quindi quando un'altra finestra diventa attiva un altro oggetto di presentazione sarà attivo.

Il controllo del flusso fra finestre o *frame* può essere rappresentato con un diagramma d'interazione (diagramma di sequenza o di collaborazione UML), mostrando quali finestre sono aperte, quale di queste è attiva e quale oggetto di presentazione è visualizzato in ogni finestra in un certo momento.

6.3.1 Elementi di Modellazione

Gli elementi di modellazione usati nel diagramma del flusso di presentazione sono: gli utenti (attori) e gli elementi di modellazione definiti nella sezione precedente, come finestre, *frame* o *frameset*.

6.3.2 Metodo

Le seguenti sono delle linee guida fornite per assistere lo sviluppatore nella modellazione del diagramma del flusso di navigazione, queste si basano sul diagramma di presentazione e su quello della struttura di navigazione.

1. Si scelga lo scenario per modellare l'interazione, cioè quale cammino di navigazione del diagramma della struttura di navigazione sarà modellato. Un cammino di navigazione è sempre relativo ad un caso d'uso.
2. Si ponga nella dimensione orizzontale l'attore, le finestre o i frame.
3. Si usi il messaggio display per ogni oggetto di presentazione che deve essere presentato all'utente (in una finestra o frame). Il parametro della chiamata "display" è proprio l'oggetto di presentazione corrispondente.
4. Si includa un messaggio select per ogni azione utente, che selezioni un'ancora od un bottone. L'ancora ed il bottone sono proprio i parametri del messaggio.
5. Si specifichi un messaggio fill o submit per ogni azione dell'utente che consiste nell'aggiungere dati in una "query form". Questa "form" è il parametro del messaggio.
6. Si includa un messaggio per ogni open e close di una finestra.
7. Si usi un "balking" per indicare il periodo di tempo in cui una finestra o un *frame* è attivo.

I diagrammi di sequenza UML sono usati per rappresentare il flusso di controllo della presentazione. Da notare che la rappresentazione non include classi aggiuntive necessarie nell'implementazione per facilitare la comunicazione fra finestre.

6.4 Diagramma del Ciclo di Vita degli Oggetti

L'obiettivo di questo diagramma è di modellare il ciclo di vita degli oggetti di presentazione reattivi, e l'influenza che hanno sullo stato degli altri oggetti di presentazione. Il ciclo di vita di un oggetto è definito da un insieme di stati e transizioni fra stati. Uno stato è caratterizzato da un nome, un'azione entrante ed un'uscente, da transizioni interne, e/o da sottostati. Una transizione che scatta al verificarsi di un determinato evento, può avere un'azione associata. Nel caso delle interfacce utente la maggior parte degli eventi sono generati dall'utente, come un "click del mouse", un tasto della tastiera che è pigiato, ecc. I comportamenti più complessi possono essere modellati in UML con dei sotto stati [40]. Si possono avere due tipi diversi di sotto stati: sequenziali e concorrenti [8].

Per rappresentare questo diagramma sono stati scelti i diagrammi di stato UML. Poiché la modellazione dei diagrammi di stato richiede molto tempo, e di solito non sono necessari per quelle classi di presentazione con comportamento già noto, sono usati solo per classi di presentazione composte e molto complesse.

6.4.1 Elementi di Modellazione

Gli elementi di modellazione usati per il diagramma del ciclo di vita degli oggetti sono gli stati e le transizioni, così come sono stati definiti in UML per costruire i diagrammi di stato.

6.4.2 Metodo

Il diagramma del ciclo di vita degli oggetti è costruito da un insieme di diagrammi di stato. Si possono usare seguenti regole come guida:

1. Si identifichino i differenti stati di un'istanza di una classe durante la presentazione (ciclo di vita dell'oggetto). Si rappresentino questi stati tramite un diagramma degli stati UML.
2. Si specifichino le transizioni fra stati determinando l'evento che provoca questa transizione.
3. Si definisca al più uno stato iniziale e, possibilmente uno o più stati finali per ogni diagramma di stato.

4. Si stabilisca se esiste qualche dipendenza fra transizioni di stato d'oggetti diversi. Le dipendenze sono rappresentate tramite da una linea tratteggiata come consiglia la notazione UML.
5. Si determini la sincronizzazione delle transizioni.

7 Diagrammi di Stato e d'Interazione per la Modellazione degli Scenari Web

Per la visualizzazione degli scenari Web, in UWE si possono usare anche i diagrammi di stato, i quali ci permettono di descrivere in modo più dettagliato parti del modello della struttura di navigazione, specificando gli eventi che provocano le transizioni, definendo le "guard condition", ed includendo esplicitamente le azioni che devono essere eseguite. Gli stati sono nominati dopo che le classi di presentazione sono visualizzate nell'interfaccia utente, infatti, il loro nome sarà assegnato dal nome della classe presentazione seguito dalla parola *displayed* (visualizzata). Ad esempio il nome dello stato sarà: "Presentational class1 displayed".

Mostrare uno scenario con questo metodo, permette di introdurre molti dettagli specifici, ad esempio si può determinare esplicitamente quando la navigazione all'indietro è consentita. Nel caso di una presentazione a finestra multipla, gli UML *synch states* sono usati per il controllo della sincronizzazione delle regioni concorrenti delle macchine a stati delle varie finestre.

I diagrammi di sequenza UML mostrano l'interazione fra oggetti, in base ad un ordine temporale, e presentano la partecipazione degli oggetti nelle interazioni e la sequenza dei messaggi scambiati tra loro.

Il metodo proposto da Conallen [9], usa i diagrammi di sequenza per la descrizione della realizzazione dei casi d'uso. Cioè come i casi d'uso sono implementati.

UWE propone i diagrammi di sequenza per raffigurare i flussi di presentazione, com'è stato già esposto sopra nel paragrafo dedicato al diagramma del flusso di presentazione.

I diagrammi di collaborazione sono equivalenti ai diagrammi di sequenza, ma diversamente da quest'ultimi i digrammi di collaborazione mostrano la relazione fra i ruoli.

Il tipo di diagramma scelto dal progettista per la descrizione degli scenari Web dipende dal grado di granularità desiderato.

8 Diagrammi d'Attività per la Modellazione dei Task

Il concetto di task è stato introdotto dallo Human Computer Interaction (HCI) field [46]: un *task* è composto di uno o più *sottotask* e/o *azioni* che un utente può eseguire per raggiungere un *obiettivo*; un obiettivo rappresenta una variazione desiderata dello stato del sistema, e può essere realizzato formulando ed eseguendo un piano (progetto) composto di *task*; le azioni sono dei *task* primitivi che non hanno struttura. In UWE si usa il concetto di *task* in modo più ampio considerando sia *task* eseguiti dall'utente (*user task*) che quelli eseguiti dal sistema (*system task*).

Sono state proposte diverse notazioni UML per la modellazione dei *task*. Wisdom è un'estensione dell'UML che propone l'uso di un insieme di classi stereotipate che rendono la notazione non molto intuitiva [34]. Markopoulos in [32] e [33] fece due proposte diverse: la prima era, un'estensione dei diagrammi dei casi d'uso UML, l'altra è basata sui diagrammi di stato ed i diagrammi d'attività. I casi d'uso dell'utente possono essere considerati come dei *task*, quando siamo ancora ad un livello d'analisi. Poiché i diagrammi d'attività sono usati usualmente per successivi raffinamenti dei casi d'uso, in UWE si usano i diagrammi d'attività per la modellazione dei *task*.

Gli elementi di modellazione per la modellazione dei *task* sono, quindi, quelli utilizzati per i diagrammi delle attività, cioè *attività*, *transizioni*, *branche*, ecc. I diagrammi d'attività in generale possono essere considerati come una "mappa" del comportamento funzionale del sistema [30]. Con l'estensione del concetto di *task* data prima, in questa sede si può parlare di "mappa" per l'interazione dell'utente con il sistema. Questa "mappa" facilita la generazione automatica d'applicazioni Web [28].

Nel modello dei *task* si usa la dipendenza stereotipata UML <<refine>> fra le attività e i diagrammi d'attività per indicare un grado d'astrazione più fine. Si sceglie anche una distribuzione verticale per rappresentare la gerarchia dei *task*, cioè da quelli modellati in modo più grossolano ai vari raffinamenti, questa rappresentazione è simile a quella usata nel "ConcurTaskTree" di [35]. L'ordine temporale fra *task* è espresso da transizioni fra attività

(attraverso i branche). La modellazione dei *task* nel campo dell'HCI comprende anche la descrizione degli oggetti – chiamati *referents* – che l'utente percepirà. Questi sono gli oggetti di presentazione e concettuali che sono introdotti nella modellazione dei *task*. Le relazioni fra i *task* e questi oggetti si rappresentano tramite un flusso d'oggetti. Si usano oggetti di presentazione entranti per esprimere l'input utente attraverso questi oggetti di presentazione e degli oggetti di presentazione uscenti per esprimere l'output mostrato all'utente attraverso questi oggetti di presentazione. In oltre, si usano oggetti concettuali per esprimere l'input e l'output dei *task*. La semantica imposta sulle espressioni dei branch nella modellazione dei *task* è che questi sono della espressioni booleane su degli oggetti del flusso d'oggetti.

9 Diagramma dei Componenti e di Distribuzione

Si usa il diagramma di distribuzione per documentare la distribuzione dei componenti delle applicazioni Web. Gli elementi principali nei diagrammi di distribuzione UML sono i *nodi*, i quali sono rappresentati graficamente tramite dei cubi. Un nodo è un elemento fisico che esiste a tempo d'esecuzione e rappresenta una risorsa computazionale [8]. Un nodo può contenere oggetti e *componenti* che risiedono all'interno della risorsa computazionale. Un componente UML è una parte, fisica e sostituibile, del sistema la quale esegue e rende possibile la realizzazione di un insieme d'interfacce, è rappresentato tramite un rettangolo con delle piccole linguette. Può essere modellata anche la connessione fisica fra nodi.

10 Costruire Applicazioni Web Adattabili

L'approccio di UWE è molto utilizzato per la costruzione d'applicazioni Web adattabili. Esse sono delle applicazioni “personalizzabili” a seconda delle preferenze o della conoscenza o degli interessi dell'utente. Nelle applicazioni Web “adattabili” queste “personalizzazioni” avvengono dinamicamente apprendendo dalla navigazione dell'utente e dal suo modo d'interagire con il sistema. Il processo adattativo può includere variazioni circa la visualizzazione di porzioni d'informazioni che sono appropriate al livello di conoscenza dell'utente, oppure dei cammini guidati che possono essere prodotti attraverso la rimozione di determinati link che il sistema considera non rilevanti per l'utente. Il sistema prende determinate decisioni in accordo alla conoscenza che il sistema ha dell'utente in un determinato periodo di tempo, conoscenza che è data dallo stato corrente del profilo utente.

Per la costruzione di applicazioni Web adattabili, UWE propone la costruzione del modello di adattamento in aggiunta dei modelli visti precedentemente.

Purtroppo però l'approfondimento delle tematiche riguardanti questo tipo di applicazioni prescindono dallo scopo di questa tesi, quindi non saranno approfondite oltre in questa sede.

In Appendice A è comunque presente il profilo UML che permette di rappresentare anche il modello di adattamento.

Un'analisi dettagliata di questa tematica è presente in [21].

Capitolo 2

IL METAMODELLO UWE³

Il metamodello per UWE è stato progettato come un'estensione conservativa del metamodello per UML. Esso è stato definito in [47], ed esteso in [27].

1 Introduzione al Metamodello

Un metamodello è una definizione precisa degli elementi di modellazione, delle loro relazioni e di regole “well-formedness” necessarie per la creazione della semantica dei modelli. I metamodelli giocano un ruolo fondamentale nella costruzione dei CASE Tool, e costituiscono il fulcro per la generazione automatica del codice.

Per estensione conservativa s'intende che gli elementi di modellazione del metamodello UML non sono stati modificati, ma tutti gli elementi del metamodello UWE, sono stati collegati tramite la relazione d'ereditarietà ad al più un elemento del metamodello UML. Inoltre, per la definizione dei nuovi elementi sono state stabilite le caratteristiche e relazioni aggiuntive e la definizione dei vincoli OCL (regole “well-formedness”) tramite i quali si definisce la loro semantica.

In figura 2.1 si può vedere che “ConceptualModel”, “NavigationModel”, “PresentationModel” sono tutte sottoclassi della metaclassa Model di UML. “ConceptualClass”, “NavigationClass” e “PresentationClass” sono tutte derivate da “UweClass”, che a sua volta è una sottoclasse della metaclassa “Class” di UML.

³ In questa tesi si fa riferimento alla versione del metamodello definito in [47].

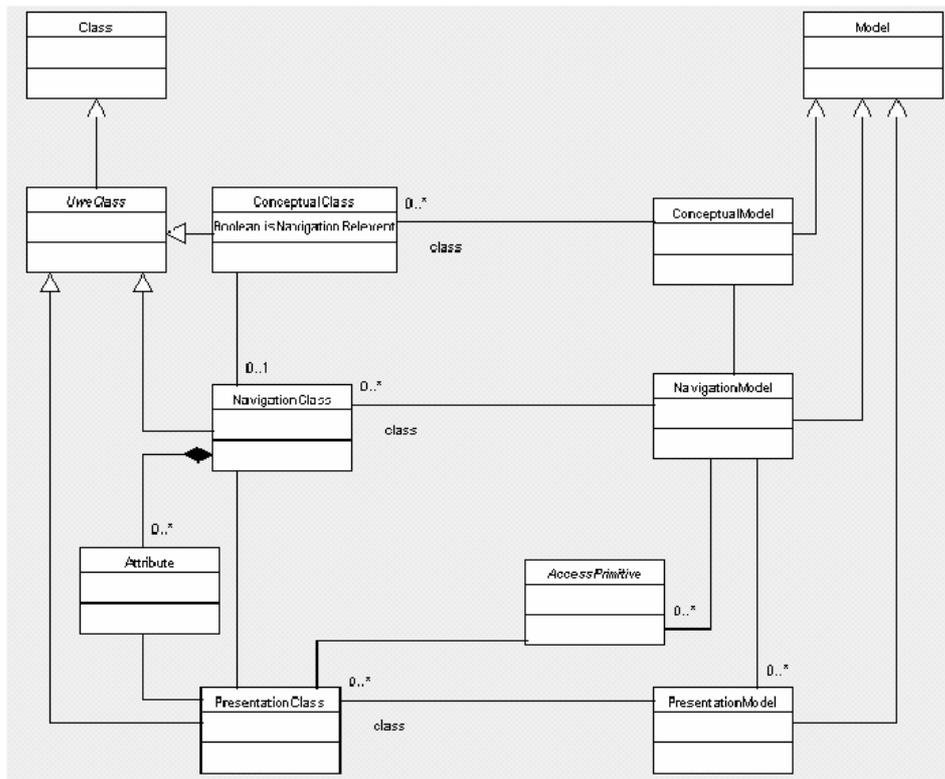


Figura 2.1. Metamodello UWE (backbone)

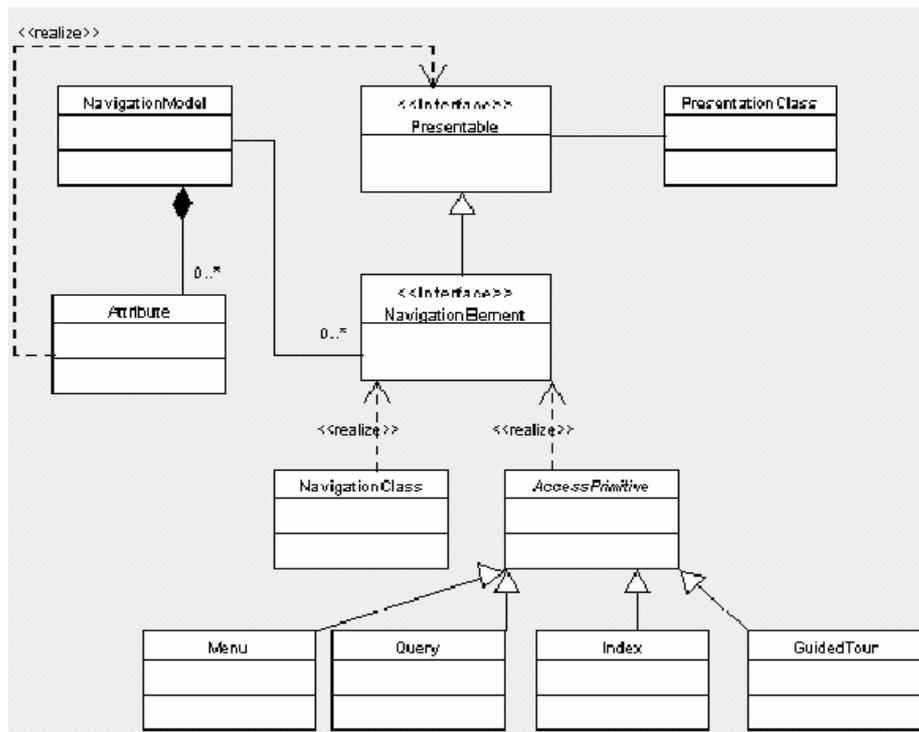


Figura 2.2. Metamodello UWE (presentable)

In questo metamodello, sono state definite anche le primitive d'accesso le quali collaboreranno con le classi di navigazione. Oltre le classi di navigazione e le primitive d'accesso, per ogni attributo delle classe di navigazione, è necessaria una classe di presentazione.

Per le classi di navigazione, primitive d'accesso e presentazione si definisce un'interfaccia: "Presentable", rappresentata in figura 2.2. Per ogni elemento presentabile ci sarà una classe di presentazione per rappresentare questi oggetti nel modello di presentazione. L'interfaccia "NavigationElement", derivata dall'interfaccia "Presentable", comprende le "NavigationClass" e le "AccessPrimitive".

Un "menu" contiene "menu item". Ogni "menu item" ha un nome ed un "link" ad un altro elemento di navigazione. Il metamodello per la classe menu è mostrato in figura 2.3. Il numero dei "menu item" in un "menu" non deve essere mai minore di uno ed in ogni caso deve essere sempre uguale al numero di associazioni uscenti dal "menu".

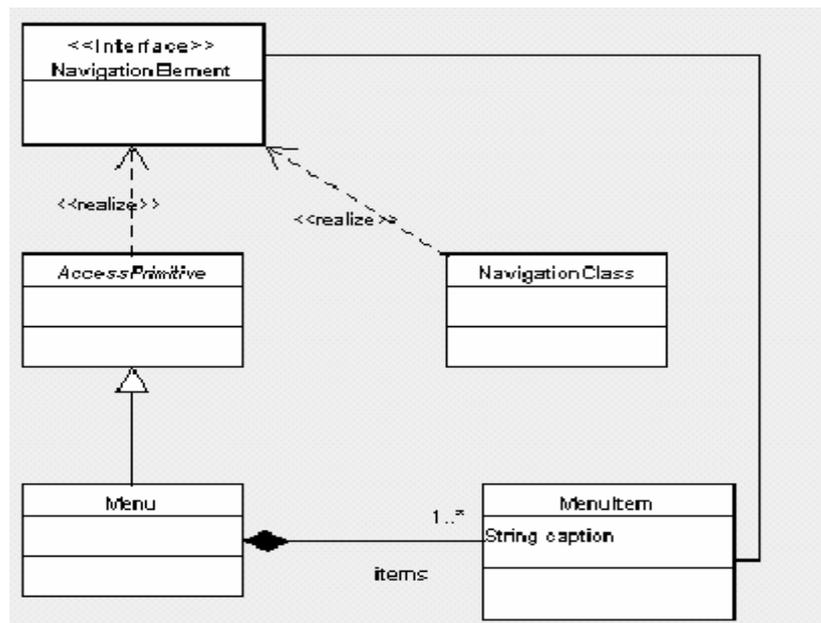


Figura 2.3. Metamodello UWE (menu)

2 Vincoli OCL per il Metamodello

Per definire il metamodello UWE, ci sono dei vincoli che devono essere definiti. In questo paragrafo ci occuperemo proprio di questo. I vincoli sono definiti in OCL.

2.1 Notazioni

Prima di iniziare, bisogna definire qualche notazione:

- a) per ogni `Model`, si definiscono un insieme finito di `association`, cioè un numero finito di associazioni nel modello:

```
context Model def:
  let association =
    self.ownedElement ->
    select(a | a.ocIsKindOf(Association))
```

- b) Si definisce, anche, un insieme di `navigationElement` per tutti gli elementi di navigazione presenti nel diagramma di navigazione, ed un insieme `presentable` per indicare un formato generico per i `navigationElement` e per tutti gli attributi delle classi di navigazione.

```
context NavigationModel def:
  let navigationElement =
    self.ownedElement ->
    select(ns | ns.ocIsKindOf(NavigationElement))

  let presentable =
    self.ownedElement ->
    select(a | a.ocIsKindOf(Attribute) and
              a.ocIsType(Attribute).owner.
                ocIsTypeOf(NavigationClass)
            ) ->
    union(self.navigationElement)
```

- c) Per ogni elemento di navigazione, si definiscono due insiemi:

```
context navigationElement def:
  let in =
    self.namespace.ocIsType(Model).association->
    select(a | (a.ocIsType(Association).
                connection->at(0).isNavigable
                implies
                a.ocIsType(Association).
                connection->at(0).type.
                ocIsType(NavigationElement) = self
            )
            and
            (a.ocIsType(Association).
                connection->at(1).isNavigable
```

```

        implies
        a.oclAsType(Association).
        connection->at(1).type.
        oclAsType(NavigationElement) = self
    )
)

let out =
self.namespace.oclAsType(Model).association->
select(a | (not a.oclAsType(Association).
connection->at(0).isNavigable implies
a.oclAsType(Association).
connection->at(0).type.
oclAsType(NavigationElement) = self
)
and
(not a.oclAsType(Association).
connection->at(1).isNavigable implies
a.oclAsType(Association).
connection->at(1).type.
oclAsType(NavigationElement) = self
)
)
)

```

L'insieme `in` è composto da tutte le associazioni che puntano ad un elemento di navigazione; mentre `out` contiene tutte le associazioni che provengono da un elemento di navigazione.

In questa definizione, bisogna tenere in considerazione che:

- § Ogni elemento di navigazione è sempre posto in un modello di navigazione; i modelli di navigazione sono una generalizzazione dei modelli (Model);
- § Esattamente un estremo dell'associazione è navigabile (vincoli 1 e 2).

2.2 Vincoli

Adesso analizziamo i vincoli:

1. Ogni associazione ha solo due estremi.

```

context Model
inv: self.association ->
    forAll(a | a.oclAsType(Association).connection
        ->size()=2)

```

Per semplicità sono ignorati i casi in cui un'associazione ha più di due estremi.

2. Nel modello di navigazione, non possono esserci delle associazioni bi-direzionali.

```

context NavigationModel
  inv: self.association ->
    forAll(a | a.oclassType(Association).
      connection->at(0).isNavigable <>
      a.oclassType(Association).
      connection->at(1).isNavigable
    )

```

Se il progettista vuole consentire la navigazione fra due elementi di navigazione, in avanti ed in indietro, dovranno essere usate due associazioni unidirezionali in direzioni opposte.

3. In ogni modello di navigazione, c'è uno ed un solo elemento di navigazione che non ha associazioni entranti. Quest'elemento di navigazione deve essere una classe di navigazione.

```

context NavigationModel
  inv: self.navigationElement->
    select(e|e.oclassType(NavigationElement).in->
      size() = 0
    )->size() = 1
  and
  self.navigationElement->
    select(e|e.oclassType(NavigationElement).in->
      size() = 0
    )->
    forAll(n|n.oclassIsTypeOf(NavigationClass))

```

Ovviamente, ogni applicazione Web deve avere un punto di partenza. Con questo vincolo, si permette solo ad una classe di navigazione di essere un punto di partenza, teoricamente questo compito potrebbe essere svolto anche dai menu. In questo caso, il progettista può sempre aggiungere una classe di navigazione che punta ad un menu, come punto di inizio di un'applicazione Web.

4. Il numero di menu item, è uguale al numero d'associazioni uscenti dal menu.

```

context NavigationModel
  inv: self.ownedElement ->
    select(e | e.oclassIsTypeOf(Menu)) ->
    forAll(m | m.oclassType(Menu).item->size() =
      m.oclassType(Menu).out->size())

```

Infatti, deve esistere un'associazione per ogni menu item e viceversa.

5. Una primitiva d'accesso ha esattamente un'associazione entrante.

```

context NavigationModel
  inv: self.ownedElement ->
    select(e|e.oclassIsKindOf(AccessPrimitive)) ->
    forAll(ap| ap.oclassType(AccessPrimitive).in->size()
      = 1)

```

Da una classe di navigazione all'altra, c'è "solo un cammino senza cicli".

6. Solo un menu può avere più di un'associazione uscente.

```
context NavigationModel
  inv: self.ownedElement ->
    forAll(c| (not c.ocIsTypeOf(Menu)) implies
      c.ocIsType(NavigationElement).out->size() <= 1)
```

Se una classe di navigazione punta a molte altre classi di navigazione, deve essere aggiunto un menu.

Query, guided tour, ed indici sono primitive d'accesso che puntano ad istanze di una classe di navigazione, quindi queste primitive d'accesso non possono puntare a diversi elementi di navigazione.

7. Una primitiva d'accesso deve avere al più un'associazione uscente.

```
context NavigationModel
  inv: self.ownedElement ->
    select(e|e.ocIsKindOf(AccessPrimitive)) ->
    forAll(ap|ap.ocIsType(AccessPrimitive).
      out->size() >= 1)
```

Un cammino di navigazione non deve finire con una primitiva d'accesso, ma sempre con una classe di navigazione.

Insieme con il vincolo 6, questo vincolo ribadisce che ogni indice, ogni query ed ogni guided tour ha esattamente un'associazione uscente.

8. Nel modello di navigazione, se la molteplicità dell'estremo di un'associazione è maggiore di un'associazione punta ad una classe di navigazione.

```
context NavigationModel
  inv: self.ownedElement ->
    select(ae | ae.ocIsTypeOf(AssociationEnd)) ->
    forAll(e | e.ocIsType(AssociationEnd).
      multiplicity.range.upper > 1 implies
      (e.ocIsType(AssociationEnd).
        type.ocIsTypeOf(NavigationClass) and
        e.ocIsType(AssociationEnd).isNavigable
      )
    )
```

Avere un'associazione con molteplicità maggiore di uno non avrebbe senso per un'associazione che punta ad una primitiva d'accesso.

9. Nel modello di navigazione, se la molteplicità di un estremo di un'associazione è maggiore di uno allora quest'associazione deve provenire da una primitiva d'accesso.

```

context NavigationModel
  inv: self.ownedElement ->
    select(ae | ae.ocIsTypeOf(AssociationEnd) and
            ae.ocIsType(AssociationEnd).
            multiplicity.range.upper > 1) ->
    forAll(e | e.ocIsType(AssociationEnd).
            association.connection->
            forAll(end | end <> e.ocIsType(AssociationEnd)
                    implies
                    end.type.ocIsKindOf(AccessPrimitive)
            )
    )
)

```

Concordemente all'associazione 8, l'estremo dell'associazione deve essere navigabile e deve puntare ad una classe di navigazione. Se ci saranno più istanze di una classe di navigazione in un sistema, si deve usare una primitiva d'accesso per organizzarle.

10. Solo una classe di navigazione può puntare ad un menu.

```

context Menu
  inv: self.in ->
    forAll(a |
      (a.ocIsType(Association).connection->
        at(0).type.ocIsType(Menu) = self
      implies
        a.ocIsType(Association).connection->
        at(1).type.ocIsTypeOf(NavigationClass))
    and
      (a.ocIsType(Association).connection->
        at(1).type.ocIsType(Menu) = self
      implies
        a.ocIsType(Association).connection->
        at(0).type.ocIsTypeOf(NavigationClass))
    )
)

```

Teoricamente un menu può avere un sottomenu, in questa versione del metamodello questo caso è ignorato.

11. Un indice o un guided tour puntano sempre ad una classe di navigazione.

```

context Index
  inv: self.out ->
  forAll(a | (a.oclAsType(Association).connection->
    at(0).type.oclAsType(Index) = self
    implies
    a.oclAsType(Association).connection->
    at(1).type.oclIsTypeOf(NavigationClass))
  and
  (a.oclAsType(Association).connection->
    at(1).type.oclAsType(Index) = self
    implies
    a.oclAsType(Association).connection->
    at(0).type.oclIsTypeOf(NavigationClass))
)

context GuidedTour
  inv: self.out ->
  forAll(a | (a.oclAsType(Association).connection->
    at(0).type.oclAsType(GuidedTour) = self
    implies
    a.oclAsType(Association).connection->
    at(1).type.oclIsTypeOf(NavigationClass))
  and
  (a.oclAsType(Association).connection->
    at(1).type.oclAsType(GuidedTour) = self
    implies
    a.oclAsType(Association).connection->
    at(0).type.oclIsTypeOf(NavigationClass))
)

```

Per definizione sia di indice sia di guided tour, non avrebbe senso che essi puntassero ad un qualsiasi altro elemento.

12. Un query non può puntare né ad un menu, né ad un'altra query.

```

context Query
  inv: self.out ->
  forAll(a | (a.oclAsType(Association).connection
    ->at(0).type = self
    implies
    (not a.oclAsType(Association).connection->
    at(1).type.oclIsTypeOf(Query)
    and
    not a.oclAsType(Association).connection->
    at(1).type.oclIsTypeOf(Menu)
    ))
  and
  (a.oclAsType(Association).connection->
    at(1).type = self
    implies
    (not a.oclAsType(Association).connection->
    at(0).type.oclIsTypeOf(Query)
    and
    not a.oclAsType(Association).connection->
    at(0).type.oclIsTypeOf(Menu)
    ))
)

```

Per la definizione di query, data nel primo capitolo, non avrebbe senso che esse puntassero ad un menu od ad un query.

13. Ogni classe concettuale rilevante per la navigazione, deve avere una classe di navigazione corrispondente.

```
context ConceptualModel
  inv: self.class -> forAll
    (c | c.isNavigationRelevant implies
      c.navigationClass -> notEmpty())
```

14. Ogni classe di navigazione deve essere derivata da una classe concettuale rilevante per la navigazione.

```
context NavigationModel
  inv: self.class -> forAll
    (c | c.conceptualClass -> notEmpty() and
      c.conceptualClass.isNavigationRelevant
    )
```

Non possono esistere classi di navigazione senza una classe proprietaria.

15. Classi concettuali rilevanti per la navigazione diverse, hanno diverse classi di navigazione.

```
context ConceptualModel
  inv: self.class ->
    forAll (con1, con2 |
      (con1 <> con2 and
        con1.isNavigationRelevant and
        con2.isNavigationRelevant
      ) implies
        con1.navigationClass <> con2.navigationClass
    )
```

16. Diverse classi di navigazione hanno diverse classi concettuali.

```
context NavigationModel
  inv: self.class -> forAll
    (nav1, nav2 |
      nav1 <> nav2 implies
        nav1.conceptualClass <> nav2.conceptualClass
    )
```

Questo vincolo insieme con i vincoli 13, 14, e 15 vuole indicare che la corrispondenza fra le classi concettuali rilevanti per la navigazione e le classi di navigazione è una funzione biettiva.

17. Ogni elemento presentabile deve avere una classe di presentazione corrispondente.

```
context NavigationModel
  inv: self.presentable ->
    forAll(np | np.oclAsType(Presentable).
      presentationClass->notEmpty())
```

Per tutti gli elementi presentabili, bisogna dare una rappresentazione visuale all'utente.

18. Elementi presentabili diversi hanno diverse classi di presentazione.

```
context NavigationModel
  inv: self.presentable ->
    forall(np1, np2 | np1 <> np2 implies
      np1.oclAsType(Presentable).
        presentationClass <>
      np2.oclAsType(Presentable).
        presentationClass
    )
)
```

19. Ogni classe di presentazione deve avere un elemento presentabile corrispondente.

```
context PresentationModel
  inv: self.class ->
    forall(pre | pre.presentable->notEmpty())
)
```

Non possono esistere classi di presentazione senza un proprietario.

20. Differenti classi di presentazione hanno differenti elementi di presentazione.

```
context PresentationModel
  inv: self.class ->
    forall(pre1, pre2 | pre1 <> pre2 implies
      pre1.presentable <>
      pre2.presentable
    )
)
```

Questo vincolo insieme con i vincoli 17, 18, e 19 vogliono indicare che la corrispondenza fra gli elementi presentabili e le classi presentazione è una funzione biettiva.

Capitolo 3

ARGOUWE – CASE TOOL PER LA MODELLAZIONE DI APPLICAZIONI WEB

ArgoUWE [2] è un CASE Tool che supporta l'approccio dell'UWE, è ancora in fase di sperimentazione ed è stato creato da Gefei Zhang⁴ [47], un ricercatore dell'Università Ludwig-Maximilians di Muchen, Germania. ArgoUWE è un'estensione di ArgoUML[1]. Di seguito saranno mostrate, le funzionalità fornite da ArgoUML, e le caratteristiche innovative fornite da ArgoUWE. ArgoUWE è stato costruito in base al metamodello UWE mostrato nel capitolo 2.

1 Introduzione ad ArgoUWE

Il CASE Tool ArgoUWE è stato sviluppato per aiutare i progettisti di applicazioni Web che utilizzano l'approccio dello UWE. ArgoUWE è un'estensione flessibile dell'ArgoUML (versione 0.10). Il cuore del CASE Tool è il metamodello UWE sottostante, definito come un'estensione conservativa del metamodello UML.

Sono stati introdotti dei nuovi diagrammi per rappresentare i nuovi tipi di modelli presentati dalla metodologia UWE: il modello concettuale, il modello di navigazione, il modello di presentazione. ArgoUWE supporta una generazione semiautomatica di questi modelli, come la generazione del modello di navigazione dal modello concettuale e la generazione del modello di presentazione dal modello di navigazione. Inoltre, ArgoUWE aiuta il progettista mantenendo il modello consistente, tramite la verifica dei vincoli definiti dalla metodologia UWE. Qualche vincolo è costantemente controllato dal *tool*, mentre per verificare gli altri si deve attendere una specifica azione dell'utente.

ArgoUWE è un tool ancora in via di sperimentazione, infatti, i suoi progettisti stanno ancora lavorando su dei piccoli miglioramenti circa l'usabilità e la migrazione verso le ultime versioni di ArgoUML. Il loro scopo è di aggiungere tutte le regole "well-formedness" di UWE nel meccanismo di "design critique" fornito da ArgoUML. Questo meccanismo di verifica dei

⁴ Esiste una nuova versione di ArgoUWE definita in [19]. In questa tesi, la documentazione relativa al tool si riferisce alla versione presentata in [47]. I diagrammi mostrati nei capitoli 5 e 6 sono stati costruiti con l'ultima versione di ArgoUWE[2].

modelli, garantisce che le regole del metamodello siano costantemente verificate, e non attende un'azione specifica dell'utente come avviene attualmente in ArgoUWE.

ArgoUWE è integrato nell'*OpenUWE tool suite environment* per permettere la generazione dei modelli guidata per qualche applicazione Web.

2 OpenUWE – Tool Suite per applicazioni Web

Il *tool suite* OpenUWE è un ambiente di sviluppo per la progettazione e la generazione di applicazioni Web. Il principio base di OpenUWE è un'architettura aperta, basata su degli *standard* ben stabiliti che sono supportati da un certo numero di *tool open source* e commerciali. Attualmente, questo ambiente di sviluppo comprende la versione alfa del CASE Tool ArgoUWE, ed un generatore di codice per lo sviluppo in un *publishing framework* XML (UWEXML). Mentre i tool per il *model checker* e per *l'editor del layout* sono ancora in via di sviluppo.

Il linguaggio usato per lo scambio dei dati all'interno di quest'architettura è definito dal metamodello di UWE il quale è stato costruito come un'estensione conservativa del metamodello UML e quindi come metamodello compatibile con il MOF (Meta Objects Facility). ArgoUWE supporta non solo la notazione UWE ma anche il modello di verifica attivo dei vincoli OCL che restringono il modello UWE. Infatti, ogni CASE Tool standard UML può anche essere usato per costruire il modello UWE, così come è stato definito dalla corrispondenza fra il metamodello ed il profilo UML di UWE.

Tecnicamente lo scambio fra i dati nei vari modelli avviene in formato XMI (XML Metadata Interchange). I modelli UWE per un'applicazione Web progettati tramite un CASE Tool UML sono completati da un modello per il layout che descrive le proprietà fisiche del layout, come colori, caratteri o dimensioni degli elementi dell'interfaccia utente. È usato un documento XML (Extensible Markup Language) per descrivere la struttura del modello del *layout* fisico per quelle proprietà fisiche degli elementi di programmazione che non sono state incluse nelle specifiche UML. Il documento XML corrispondente è prodotto dall'*editor del layout* e quindi inserito nel generatore di codice per essere trattato insieme con il modello di presentazione espresso in UML per generare il documento per la presentazione fisica per una

ben indicata tecnologia Web, come i documenti XML e XSLT (Extensible Stylesheet Language Transformations) o le pagine HTML(Hypertext Markup Language).

3 ArgoUML

ArgoUWE è un'estensione di ArgoUML, eredita quindi tutte le caratteristiche di quest'ultimo. Per uno studio più approfondito del CASE Tool in esame è necessario conoscere innanzi tutto ArgoUML, per questo motivo è analizzato in dettaglio in questo paragrafo.

ArgoUML è un CASE Tool in corso di sviluppo presso l'Università della California, il cui scopo è fornire un valido supporto all'analisi e alla progettazione di sistemi software Object-Oriented. È un progetto *open-source* che coinvolge un centinaio di persone tra ricercatori e studenti.

ArgoUML è basato direttamente sulla specifica UML 1.3 [44], infatti, una larga parte di esso è stata generata automaticamente dalla specifica stessa. Inoltre è l'unico CASE Tool che implementa il metamodello UML esattamente come è specificato.

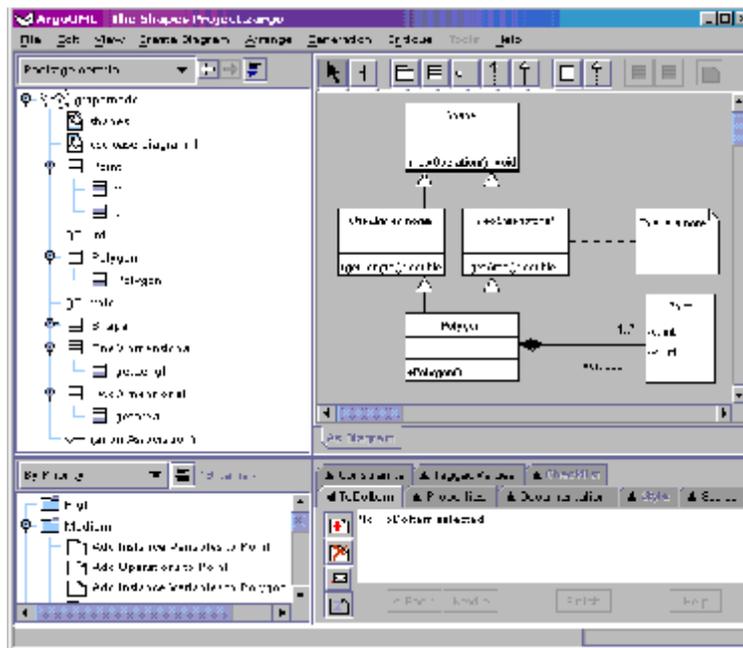


Figura 3.1: Screenshot di ArgoUML

Come è noto, le attività di analisi e di progettazione di un sistema software sono tra le più complesse e difficili. La maggior parte dei CASE Tool commerciali fornisce strumenti che riducono al minimo il lavoro manuale necessario a completare queste attività e a trasformare un progetto in codice. L'aiuto fornito dai CASE Tool maggiormente utilizzati solitamente consiste nell'offrire il supporto per le notazioni più utilizzate, per la generazione di codice, per il controllo delle versioni e altro ancora. Tuttavia, solitamente questi tool non forniscono un supporto per un determinato processo di sviluppo, quindi non offrono a chi sviluppa software un aiuto nel prendere decisioni progettuali: solitamente presentano una pagina bianca iniziale e consentono all'utente di accedere da subito a tutte le funzioni disponibili, lasciandogli il compito di completare le varie fasi del processo di sviluppo. Al contrario, ArgoUML, pur non disponendo di tutte le funzionalità dei principali CASE Tool commerciali, si distingue per le sue caratteristiche innovative nel guidare gli sviluppatori attraverso le varie fasi di un processo di sviluppo iterativo che si ispira al Rational Unified Process [37]. Infatti, il processo adottato è costituito da quattro fasi che si possono ricondurre a quelle del Rational Unified Process:

- prima fase (***inception*** nel Rational Unified Process): è la fase di analisi dei requisiti. Mediante i diagrammi dei casi d'uso si cerca di individuare tutte le funzionalità che il sistema deve offrire. I documenti prodotti in questa fase prendono il nome di ***vision documents***
- seconda fase (***elaboration***): viene elaborata un'architettura di base del sistema utilizzando diagrammi delle classi e dei package. Inoltre, sulla base dei requisiti individuati nella fase precedente, si utilizzano i diagrammi UML di sequenza e degli stati per descrivere il comportamento del sistema negli scenari d'uso più complessi.
- terza fase (***construction***): utilizzando i diagrammi dei package si cerca di individuare i componenti che formano il sistema. I diagrammi prodotti durante la fase precedente vengono rivisti e rielaborati, al fine di definire nel dettaglio l'architettura complessiva del sistema. Per fare questo, possono tornare utili i diagrammi UML di collaborazione, di sequenza e d'attività.

Infine, vengono utilizzati i diagrammi UML di dislocamento per descrivere il sistema reale.

- quarta fase (***deployment***): è la fase di codifica del sistema. UML non è stato progettato per fornire un valido sostegno durante questa fase. Il supporto fornito da ArgoUML consiste nel consentire di generare codice ***stub*** dai diagrammi delle classi e dei package.

Anche utilizzando Rational Rose è possibile applicare il Rational Unified Process. Dal punto di vista materiale, esso si presenta come una guida in linea che illustra in maniera dettagliata i ruoli, le attività, gli elaborati che occorre produrre, con l'aiuto di linee guida, template per

documenti, esempi. Poiché non sono disponibili strumenti specifici per integrare Rational Rose con il Rational Unified Process, non viene imposto nessun vincolo agli sviluppatori che intendono adottare questo processo di sviluppo utilizzando Rational Rose. Al contrario, ArgoUML impedisce di completare alcune attività se altre non sono state portate a termine. Occorre tuttavia notare che in alcuni casi questo può risultare piuttosto restrittivo e limitante.

3.1 Perché ArgoUML è diverso

Coloro che hanno familiarità con gli strumenti per l'analisi e la progettazione di sistemi software Object-Oriented, troveranno ArgoUML subito familiare. Tuttavia ArgoUML si differenzia dalla maggior parte dei CASE Tool tradizionali per alcune importanti caratteristiche che possono essere riassunte in quattro punti:

- Applicazione di principi derivanti dalla psicologia cognitiva.
- Impiego esclusivo di tecnologie standard.
- 100% Pure Java.
- Open source.

Psicologia cognitiva

ArgoUML mette in pratica studi teorici nel campo della psicologia cognitiva, allo scopo di fornire strumenti innovativi per aumentare la produttività e supportare al meglio il processo di sviluppo del software nelle fasi di analisi e progettazione Object-Oriented. Come detto, ArgoUML si propone di guidare lo sviluppatore attraverso le varie fasi di sviluppo di un processo simile al Rational Unified Process. Per ottenere questo, vengono applicati i principi di tre differenti teorie nel campo della psicologia cognitiva:

- **Reflection-in-action:** questa teoria trae spunto dall'osservazione che, quando si inizia un'attività di progettazione di un sistema complesso da un punto di vista architettonico, non si riesce a sviluppare da subito un'idea dell'architettura complessiva del sistema. Infatti, solitamente viene elaborato un primo progetto parziale, il quale viene valutato e rielaborato, riflettendoci sopra e confrontando le idee delle altre persone coinvolte nello sviluppo del sistema, finché si ritiene di essere in grado di estenderlo e migliorarlo. Nell'ambito dell'ingegneria del software, questa considerazione ha contribuito alla nascita dei modelli di sviluppo iterativi, come lo stesso Rational Unified Process, che si propongono come alternativa al tradizionale modello a cascata. La psicologia cognitiva definisce questa attività di apprendimento come il processo di “dare un senso alla

complessità”, ossia la riflessione ed il confronto con altre persone per rivedere il processo di comprensione della complessità.

- **Opportunistic design:** questa teoria osserva che solitamente i progettisti software si propongono di pianificare il proprio lavoro in maniera ordinata e gerarchica, ossia stabilendo un ordine tra i lavori da eseguire. Tuttavia, questo ordine solitamente non viene rispettato, poiché si sceglie di procedere nello sviluppo del sistema scegliendo tra i compiti da portare a termine quello mentalmente meno costoso.
- **Comprehension and problem solving:** la progettazione di sistemi software Object-Oriented è un'attività complessa e la progettazione di sistemi basati su componenti riusabili lo è ancora di più. Per questo motivo, un progettista software può trarre beneficio dalla possibilità di riconoscere schemi progettuali ricorrenti, che risolvono specifici problemi progettuali e rendono l'architettura del sistema più flessibile, elegante e soprattutto riusabile. Anche i programmatori possono trarre vantaggio da questo, per comprendere meglio il sistema e rendere più agevole la sua codifica. Uno schema progettuale ricorrente, può essere un *design pattern* [14], un *framework*, o semplicemente uno stereotipo [38].

I principi della psicologia cognitiva trovano applicazione in ArgoUML mediante i seguenti aspetti che lo caratterizzano:

1. Un'interfaccia utente che permette di esaminare i documenti di un progetto da diverse prospettive, permettendo di perseguire un obiettivo percorrendo diverse strade alternative. Infatti, ArgoUML, come molti altri CASE Tool, fornisce un menu grafico organizzato gerarchicamente (figura 3.2), che permette di accedere alle varie parti del progetto, come un diagramma UML o un singolo elemento di un diagramma. Tuttavia, al contrario dei tool tradizionali, ArgoUML consente di organizzare il menu secondo diverse alternative, permettendo di esaminare un progetto da numerose prospettive differenti. Per esempio, è possibile organizzare il menu evidenziando gli elementi di un singolo diagramma, oppure quelli dell'intero progetto, o ancora di evidenziare solo le relazioni che intercorrono tra i vari elementi.

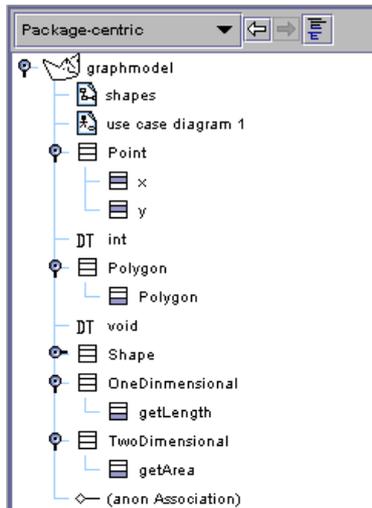


Figura 3.2: Il menu grafico di ArgoUML

2. L'utilizzo di semplici agenti software, eseguiti in *background*, che analizzano lo stato di avanzamento del progetto e forniscono all'utente suggerimenti utili. Questi riportano errori sintattici, offrono consigli sullo stile utilizzato nel disegnare i diagrammi UML, suggeriscono ottimizzazioni progettuali, come l'indicazione di utilizzare un certo design pattern o un determinato stereotipo (figura 3.3).

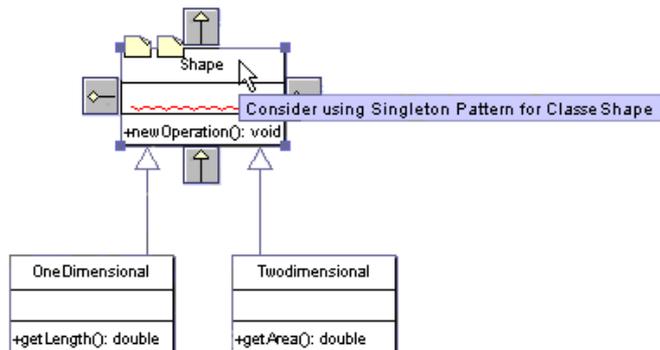


Figura 3.3: Un suggerimento progettuale

3. L'utilizzo di "to-do" list, ossia di elenchi di "cose da fare", composti da consigli progettuali, errori da correggere, note personali inserite dall'utente. Ogni elenco è personalizzabile, infatti, è possibile decidere quali elementi inserire e come organizzarli (figura 3.4).

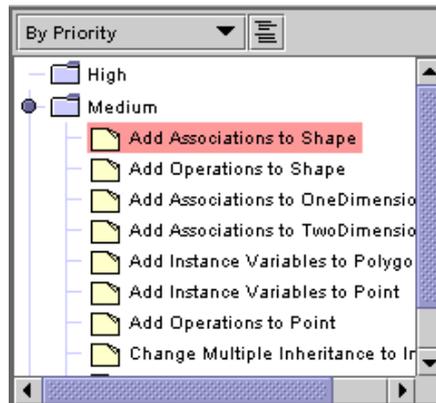


Figura 3.4: Una “to do” list

Tecnologie standard

ArgoUML utilizza esclusivamente tecnologie standard basate su XML per mantenere le informazioni relative ai diagrammi UML: XMI, PGML e SVG. Ne deriva l'indubbio vantaggio di avere interfacce comuni che favoriscono lo scambio di informazioni tra applicazioni differenti. Attualmente, questo approccio così flessibile non è condiviso dal resto dei CASE Tool commerciali, i quali utilizzano una propria rappresentazione interna dei dati. Nell'ottica di una strategia di mercato monopolistica, l'utilizzo di interfacce non standard rappresenta sicuramente il tentativo operato da un'azienda produttrice di software di tenere legati a sé i propri clienti. Un obiettivo comune dei prodotti open source è quello di evitare il controllo monopolistico sul software. L'utilizzo di tecnologie standard rappresenta quindi una scelta naturale per i prodotti open source, incluso ArgoUML.

Le caratteristiche salienti degli standard citati sono elencate di seguito:

- **XMI (XML Metamodel Interchange):** è lo standard utilizzato per salvare le informazioni relative ad un modello UML. In linea di principio, questo consente di importare un modello creato con ArgoUML all'interno di altri tool che supportano il formato XMI. Purtroppo, nella realtà questo non è sempre possibile, per due ragioni: in primo luogo XMI è uno standard recente e ArgoUML è tra i primi tool ad adottarlo, inoltre XMI non fornisce il supporto per salvare le informazioni grafiche relative ad un modello UML. Come detto, ArgoUML risolve questo problema disaccoppiando le informazioni grafiche relative ai diagrammi da quelle concernenti il modello. Le prime vengono salvate in formato PGML, per le ultime viene utilizzato

XMI. ArgoUML utilizza la versione 1.0 di XMI, che fornisce completo supporto per UML 1.3.

- **PGML (Precision Graphics Markup Language)** e **SVG (Scalable Vector Graphics)**: sono due linguaggi di grafica vettoriale per il Web proposti al World Wide Web Consortium da Adobe Systems. PGML è stato proposto come nota il 10 aprile 1998. Dopo poco tempo è stato sostituito da SVG, proposto per la prima volta al W3C nel febbraio del 1999 e divenuto una raccomandazione il 4 settembre 2001. Attualmente ArgoUML utilizza il formato PGML per salvare le informazioni grafiche relative ai diagrammi UML. Tuttavia fornisce la possibilità di esportare le stesse informazioni nel formato SVG.

100% Pure Java

ArgoUML è scritto interamente in Java 1.2. Il motivo di questa scelta è dovuto in primo luogo alla portabilità del linguaggio.

Al fine di rendere ArgoUML il più possibile indipendente dal sistema operativo sottostante, è stato utilizzato il toolkit JFC (Java Foundation Classes) [42], sviluppato da Sun Microsystem, che consente di creare interfacce utente il cui aspetto grafico è del tutto indipendente da sistema operativo utilizzato.

Open source

ArgoUML è un prodotto open source. Tutti possono ottenere una copia gratuita del codice sorgente, modificarlo e sottoporre gli aggiornamenti apportati al giudizio della comunità degli sviluppatori di ArgoUML.

4 Automazione del processo di modellazione tramite ArgoUWE

I sei passi principali del processo di modellazione di UWE, come visto nel primo capitolo, sono:

1. ***Analisi dei requisiti.*** Ovviamente questo passo non potrà essere eseguito automaticamente da un CASE Tool, ma il diagramma relativo, cioè il diagramma dei casi d'uso, verrà costruito dall'utente così come verrebbe costruito da un CASE Tool per UML.
2. ***Creazione del modello concettuale.*** L'utente deve trovare le classi, definire le eventuali ereditarietà e specificare i vincoli. ArgoUWE per far ciò dispone di un'interfaccia grafica. E' molto importante che l'utente marchi ogni classe concettuale come "navigation relevant" o

“navigation irrelevant”. ArgoUWE visualizzerà questi due tipi di classi concettuali in modo diverso.

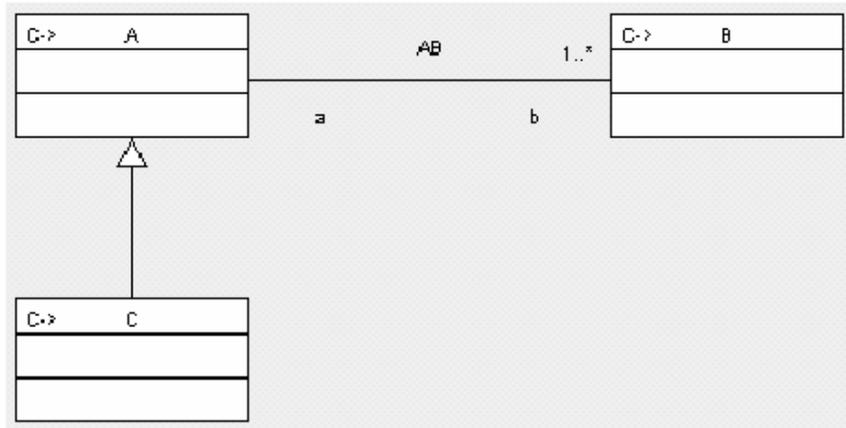
3. **Creazione del modello di navigazione.** In questo passo ArgoUWE aggiunge automaticamente, per ogni classe concettuale marcata come “navigation relevant”, una classe di navigazione (si veda il vincolo 13) ed una nuova associazione, per ogni associazione già esistente fra due classi concettuali di questo tipo. ArgoUWE crea automaticamente il diagramma di navigazione, selezionando dal menu “create diagram” la voce “Navigation Diagram”, a partire da un diagramma concettuale dal quale si vogliono ottenere le informazioni da copiare.

Nel modello di navigazione, sono modellati anche i cammini di navigazione fra gli elementi di navigazione, le relazioni di generalizzazione non rappresentano un'informazione importante per la navigazione. ArgoUWE, quindi, non trasferirà le generalizzazioni dal modello concettuale al modello di navigazione. Tutti gli attributi della classe concettuale padre saranno copiati nelle classi di navigazione delle classi concettuali figlie⁵.

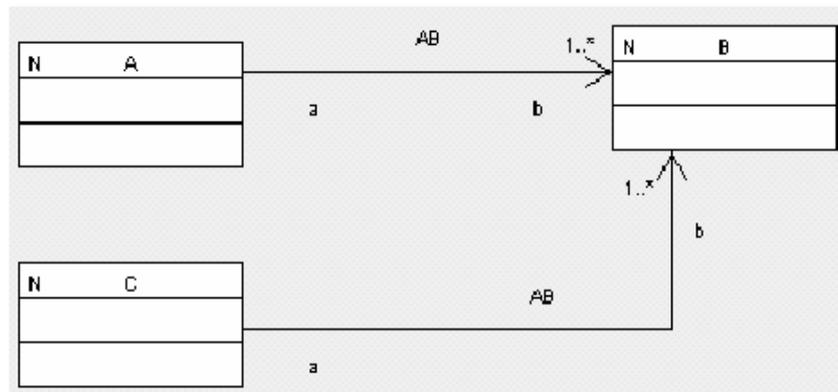
Se la classe padre A in una generalizzazione è associata con una classe B, allora C, la classe figlia di A, è anche associata con B per ereditarietà. Per ogni associazione fra A e B, ArgoUWE può aggiungere un'associazione corrispondente fra C e B se l'utente seleziona la *checkbox* “asso.inheritance”, altrimenti questo tipo di associazioni non saranno aggiunte automaticamente e quindi le dovrà aggiungere l'utente.

Un esempio di questa situazione è dato in figura. ArgoUWE non cambia il nome delle associazioni create per le classi figlie. L'utente dovrà fare lo stesso.

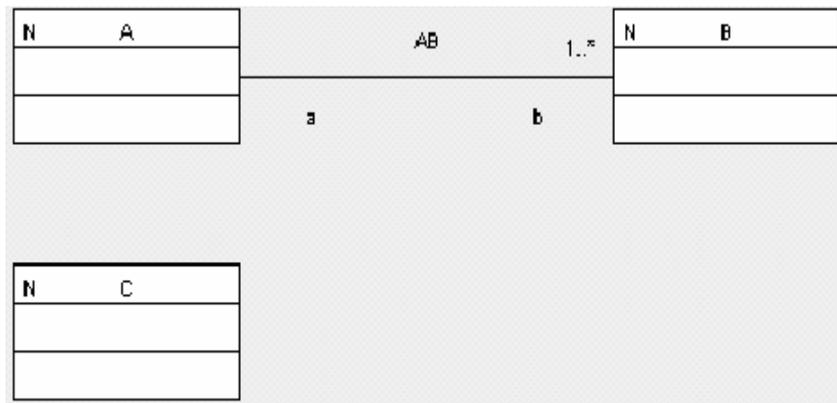
⁵ Le operazioni non sono importanti nel modello di navigazione.



Generalizzazione nel modello concettuale



Modello di navigazione se è stata selezionata "asso.inheritance"



Modello di navigazione se non è stata selezionata "asso.inheritance"

Figura 3.6: Esempio di Ereditarietà

4. **Aggiungere ulteriori associazioni.** Solo l'utente può determinare dove sono necessari degli ulteriori accessi diretti. ArgoUWE controlla, in ogni caso, le associazioni aggiunte dall'utente ed assicura che siano tutte unidirezionali. (vincolo 2).

5. **Aggiungere le primitive d'accesso.** In questo passo, ArgoUWE può aiutare l'utente ad aggiungere le primitive d'accesso in tutti i posti in cui le primitive d'accesso sono necessarie (vincoli 6 e 9).

Infatti ArgoUWE può indicare tutti:

- le associazioni che hanno una molteplicità maggiore di uno, ed aggiungere le primitive d'accesso ad esse;
- gli elementi di navigazione che hanno più di un'associazione uscente ed aggiungere un menu corrispondente.

ArgoUWE non consente all'utente di aggiungere un menu in un posto errato (vincoli 6 e 10), infatti esso prevede l'inserimento di un elemento menu, nel menu *pop up*, solo quando l'utente clicca con il tasto destro del mouse su una classe di navigazione che ha molte associazioni uscenti.

6. **Creazione del modello di presentazione.** In questo passo, ArgoUWE deve aggiungere per ogni elemento di presentazione una classe di presentazione (vincolo 17). Le classi di presentazione adiacenti saranno connesse tramite composizione. L'utente deve solamente selezionare selezionando dal menu "create diagram" la voce "Presentation Diagram", a partire da un diagramma di navigazione rappresentante il modello di navigazione dal quale si deve copiare.

4.1 Verifica della Consistenza

Una funzionalità molto importante per un CASE Tool, è quella della verifica della consistenza dei modelli di un progetto introdotto dall'utente. Se in questo progetto qualche vincolo del metamodello è violato, il CASE Tool produrrà un messaggio d'errore per avvisare l'utente.

Un CASE Tool può garantire la consistenza di un progetto in due modi:

1. controllando ogni azione dell'utente e quindi proibendo tutte quelle che non rispettano i vincoli;

2. ogni azione all'utente è permessa, però esiste una funzione che verifica se il progetto è consistente.

ArgoUWE adotta una strategia mista. Esso assicura che alcuni vincoli siano sempre soddisfatti, ad esempio non è consentito ad un utente di aggiungere un menu dopo una primitiva d'accesso (vincolo 10).

In ogni modo, non esistono vincoli che “possono” essere violati durante il processo di modellazione. Per esempio, in ArgoUML (e quindi in ArgoUWE), se l'utente desidera cambiare la direzione di un'associazione unidirezionale, deve prima rendere l'associazione bidirezionale, quindi violare il vincolo 2 per un periodo di tempo, ed inseguito indicare la nuova direzione.

4.2 Esportazione dei modelli nello standard XMI

Per implementare ArgoUWE, è stato esteso il metamodello di UML. Il nuovo metamodello potrebbe, naturalmente, non essere compreso dagli altri tool creati per UML. ArgoUWE necessita quindi di una funzione che rende possibile l'esportazione dei modelli creati dall'utente ad uno standard XMI usando un meccanismo di estensione standard dell'UML per permettere la cooperazione con altre applicazioni UML.

Per implementare questa funzionalità, ArgoUWE utilizza come meccanismo d'estensione i “tagged value”. Gli stereotipi, sono anch'essi dei meccanismi d'estensione, ma l'uso degli stereotipi è un po' più complicato dei “tagged value”.

Le regole di corrispondenza fra le metaclassi specifiche di UWE ed i relativi “tagged value” in UML standard sono mostrate nell'Appendice B.

5 Cosa non è stato implementato in ArgoUWE

ArgoUWE è un tool di modellazione. A tempo di modellazione di solito non è possibile dire esattamente quante istanze di una certa classe ci saranno a tempo d'esecuzione, per questo motivo l'“IndexItem” non è stato implementato. Il progettista non può dire esattamente quanti passi ci saranno in un guided tour, quindi nessun dettaglio dei “Guided Tour” è stato implementato.

Durante il processo di modellazione, non sempre si è in grado di decidere il layout di una classe di presentazione o come un attributo di una classe di navigazione dovrebbe essere presentato, e neanche si è in grado di dire quale classe dovrebbe essere presentata ed in quale frame. Per questo i progettisti di ArgoUWE hanno deciso di non supportare il modello della struttura di presentazione, il modello del flusso di presentazione ed il modello del ciclo di vita degli oggetti così come sono stati definiti nel capitolo 1.

Capitolo 4

AMBIENTE DI SVILUPPO E APPLICAZIONI INTEGRATE

Lo sviluppo di questa tesi è stato svolto presso la divisione CMG (Consumer Microcontroller Group) della STMicroelectronics di Catania. In questo capitolo sarà descritta la struttura hardware e software utilizzata per lo sviluppo dei casi di studio “IP Exchange” e “DVD-STB Web” che saranno descritti dettagliatamente nei capitoli 5 e 6.

1 Struttura dell'ambiente di sviluppo

Prima di parlare della progettazione e dello sviluppo dei casi di studio è necessario introdurre l'ambiente in cui si è lavorato. In particolare in questo capitolo saranno introdotte la struttura dell'*hardware* usato, del software presente e di quello successivamente introdotto per necessità implementative, e la struttura di applicazioni Software, già esistenti all'interno della STMicroelectronics, che sono state integrate nelle applicazioni “IP Exchange” e “DVD-STB Web”.

Nella descrizione dell'ambiente di sviluppo, sarà seguito l'ordine rappresentato in figura 4.1 (dal basso verso l'alto).



Figura 4.1 Componenti necessari per lo sviluppo delle applicazioni “DVD-STB Web” ed “IP Exchange”

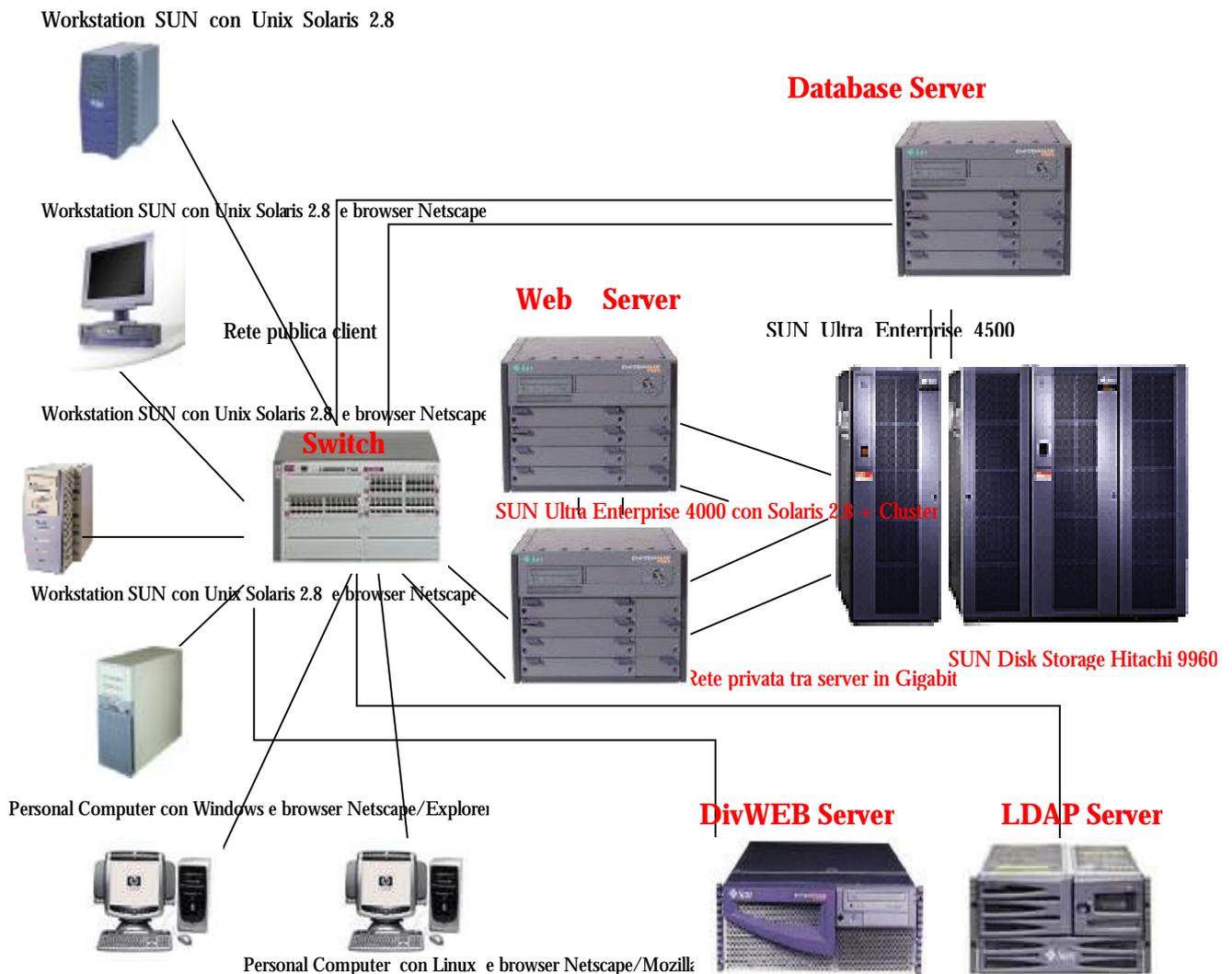


Figura 4.2. Architettura Hardware

“IP Exchange ” e “DVD-STB Web” sono delle applicazioni Web. Un’applicazione Web estende il concetto di sito Web, poiché essa permette all’utente d’interagire e cambiare lo stato dei dati presenti nel *server*. E’ un sistema software *dient/server* che ha almeno tre componenti significative da un punto di vista dell’architettura: un *browser HTML/XML*, presente su uno o più *dient*, che comunicano tramite il *server Web* attraverso il protocollo HTTP, ed un *server di applicazione* il quale è responsabile dell’esecuzione dell’applicazione, che può essere localizzato sulla stessa macchina del server Web. Molto

spesso oltre al server di applicazione esiste anche il **server di base di dati** utilizzato per rendere permanenti i dati necessari per l'esecuzione dell'applicazione.

1.1 Architettura Hardware

Come si può vedere in figura 4.2, vi sono diversi tipi di client che si collegheranno al server Web. Essi utilizzano browser diversi, quindi una caratteristica molto importante richiesta alle nostre applicazioni è la compatibilità con la maggior parte dei *browser* esistenti.

Client

1. Workstation SUN:

- Sistema operativo: Unix Solaris 2.8, 2.7, 2.6, 2.5;
- Browser: Netscape.

2. Personal Computer

- Sistema Operativo: Windows;
- Browser: Netscape/Explorer.

3. Personal Computer

- Sistema Operativo: Linux ;
- Browser: Netscape/Mozilla.

Server Web

Host Name: MCDD59.

IP Address: xxx.yyy.zzz.kk.

System Configuration : Sun Microsystem sun4u Ultra Enterprise 4000 UltraSPARCII.

System clock frequency: 336 MHz (12 CPU).

Memory size: 12 Gigabytes.

Database Server

- System Configuration: Sun Ultra Enterprise 4500;
- Disk Storage: SUN Disk Storage Hitachi 9960.

Server delle applicazioni integrate

DivWeb Server e Ldap Sever, sono dei database accessibili dalle applicazioni in studio per ricevere dati. Non è possibile indicare nessuna caratteristica circa l'hardware e il software presenti.

1.2 Architettura Software del Web Server

Sistema operativo:

1. Unix Solaris 2.8.

Librerie aggiunte per lo sviluppo delle applicazioni Web richieste :

1. Libpng 1.2.4 - utilizzata per leggere immagini di tipo png.
2. Bison 1.875 - Linker per librerie cc di GNU.
3. Libgcc - libreria di sistema per il compilatore cc (GNU cc).
4. Gdbm 1.8.3 -librerie di sistema per il compilatore gcc di GNU cc.

Compilatori:

1. Gcc 3.2.3 - compilatore cc di GNU.
2. J2SDK 1.4.1_02 - compilatore java.
3. Perl 5.0.8 – interprete perl.

Servizi:

1. Apache 1.3.26 - server Web.
2. OpenLdap 2.1.21 - server ldap.
3. SunCluster 3.0 - software che garantisce continuità e mette in alta affidabilità i server e le loro applicazioni. Un cluster è costituito da un **server master** ed uno o più **slave** e da un software di **dustering**. Il software permette di ridondare un intero **file server** costituendone una macchina gemella che fornirà gli stessi servizi della prima, in modo che per un qualsiasi problema del server master (rotture hardware o **crash** improvvisi), tutti i servizi forniti dal master saranno migrati sullo slave, in modo trasparente all'utente che usa il servizio.
4. Samba 2.2.2 – software per la condivisione dei filesystem Unix in rete anche su sistemi Windows.

Moduli Apache Aggiuntivi:

1. Mod_auth_ldap – modulo Ldap per Apache.
2. Mod_perl 1.21 – modulo perl per Apache.
3. Jpgraph 1.12.1 – modulo che consente di creare grafici e statistiche in PHP4.

4. Apache Ant 1.5.3 – Editor Java.

Linguaggi di Programmazione usati:

1. PHP4 4.3.1
2. Perl 5.8.0.
3. HTML 4.
4. Javascript.
5. DHTML.
6. C-shell script.
7. Mysql 3.23.49.

Lo sviluppo delle applicazioni è stato effettuato su una Sun Ultra 60 con *sunpci* integrata. Una *sunpci* è una scheda pc integrata in una *workstation* SUN, essa possiede un sistema operativo Windows 2000 dentro un'immagine disco C:\>, allocata nel file unix "C.diskimage". I file system di Unix Solaris sono accedibili dal PC tramite in server Samba.

Su questa macchina sono presenti i seguenti sistemi operativi:

1. Unix Solaris 2.8.
2. Windows 2000.

I "tool" utilizzati sono:

1. Macromedia MX Studio (Windows).
2. Fireworks Mx.
3. Flash Mx.
4. Dreamweaver MX.
5. Apache Ant.
6. ArgoUWE.

1.3 Linguaggi usati

Lo sviluppo delle applicazioni è stato effettuato essenzialmente tramite il linguaggio script PHP4[10] ed il RDBMS (Relational Data Base Management System) Mysql [31].

Come linguaggio script è stato scelto PHP4, perché esso è un linguaggio script per il Web, che si caratterizza per essere indipendente dal *browser* e multiplatforma, *HTML-embedded*, *lato-server* e *open-source*

Le caratteristiche principali di PHP4, sono:

- Multiplatforma. Il codice PHP4 può essere interpretato da qualunque browser, senza alterazioni, e da qualunque computer con diversi sistemi operativi.
- HTML-embedded. Il codice di PHP4 è scritto in file contenenti una miscela di istruzioni PHP4 e di codice HTML.
- Lato-server. I programmi PHP4 sono interpretabili su un Web server.
- Linguaggio di script per il Web, i programmi scritti tramite PHP4 sono interpretabili tramite browser.
- Open Source.
- Si interfaccia facilmente con molti RDBMS tra cui Mysql.
- Facile da usare.
- Possibilità di costruire pagine Web dinamiche.
- Possibilità di gestire file, database, oggetti, ecc.

Inoltre PHP4 è un linguaggio script Object Oriented, che da la possibilità di includere file “.inc” all’interno di file “.php”. Quando il file è incluso, il codice eredita lo scope (contesto all’interno del quale la variabile è definita) delle variabili della riga in cui si verifica l’inclusione.

Il codice prodotto, è una combinazione di istruzioni PHP4 e HTML si usa il primo per controllare e formattare il secondo, in modo da essere compatibile con i vari browser.

Per immagazzinare i dati in modo permanente si è scelto l’utilizzo di database relazionali, essi permettono di organizzare grandi quantità di dati, in modo tale che si possa accedere facilmente ai loro contenuti e manipolarli.

Come RDBMS, Data Base Management System, cioè il software usato per memorizzare, recuperare e modificare i dati in un database, è stato scelto Mysql.

Si è scelto Mysql perché offre:

- Ottime prestazioni e affidabilità. Mysql è notevolmente veloce e affidabile.
- MultiPiattaforma, può essere installato su diverse piattaforme (Unix, Windows).
- Facilità d’uso. Mysql è un sistema di gestione dei database relativamente semplice, nonostante le sue straordinarie caratteristiche. E’ facile da installare e gestire; è provvisto

di una serie completa di applicazioni client/server, più una serie di “utilità” che rendono l’amministrazione del database particolarmente facile.

- Compatibile con SQL.
- Si interfaccia con molti linguaggi script tra cui PHP.
- Software Open Source.

2 Applicazioni Software

Uno degli scopi principali delle applicazioni descritte nei capitoli 5 e 6 è utilizzare al meglio le risorse software già esistenti all’interno della STMicroelectronics. A tal fine è stata, effettuata un’attenta analisi delle risorse software disponibili, dalla quale sono emerse delle applicazioni che necessariamente dovevano essere integrate con quelle da me sviluppate, per soddisfare pienamente le richieste.

Nei capitoli 5 e 6, verrà spiegato il motivo delle suddette scelte, in questo paragrafo descriveremo semplicemente le loro caratteristiche. Le applicazioni in esame saranno integrate nell’applicazione “DVD-STB Web”, mentre “IP Exchange” utilizzerà solo l’“Enterprise Directory”.

2.1 DivWeb

DivWeb [12] è un sistema di database di documenti accessibili, tramite *intranet*,⁶ dalle divisioni appartenenti a CMG.

I contenuti di questi database riguardano materiale e documentazione tecnici, quale ad esempio: Report di meeting, Tool User guide, Design Flow, ecc.

Lo scopo principale è di migliorare la condivisione delle informazioni all’interno delle divisioni e di semplificare il processo della pubblicazione dei documenti (*cross fertilization, knowledge sharing*), o delle attività interne. I documenti interni alle divisioni, per motivi di sicurezza, non sono visibili all’esterno della *intranet*.

DivWeb fornisce quattro funzioni essenziali: Add, Modify, Delete, Search.

⁶ Uno intranet è una rete locale di computer (LAN), con una caratteristica particolare: il software utilizzato per i servizi di rete è lo stesso che si usa in Internet.

- **Add (Aggiunta di un nuovo Documento).** Si possono inserire documenti appartenenti a diverse categorie e tipi di documenti. I campi sono personalizzabili per ogni divisione. Un documento può essere fisicamente trasferito sul server oppure referenziato attraverso un link ad un URL che si trova in un server diverso.
- **Modify.** Un documento depositato in precedenza può essere aggiornato. Solo il proprietario di un record del database può modificare un determinato record.
- **Delete.** Un documento depositato in precedenza può essere cancellato. Solo il proprietario di un record del database può cancellare un determinato record.
- **Search.** La funzione Search estrae solo quei record rilevanti per l'input dato per la ricerca. Una stringa inserita nel campo di ricerca è ricercata negli attributi del documento. La funzione search supporta la ricerca attraverso determinati attributi ed anche l'utilizzo di operatori logici come AND ed OR.

DivWeb fornisce un servizio di “clean_up” automatico del *database*. Infatti, ogni documento presente nel database ha associata una validità decisa dal proprietario al momento dell'inserzione. In automatico la validità del documento è settata a sei mesi, ma l'utente può scegliere fra uno, due, tre, sei, dodici mesi o permanente. Il sistema settimanalmente controlla lo stato dei documenti e spedisce quindi un e-mail al proprietario quando il suo documento ha raggiunto la data di scadenza. L'utente verrà avvertito per tre volte, e se nessuna azione verrà compiuta dal proprietario, il *record* e quindi il documento verrà cancellato dopo quattro settimane dalla data di scadenza.

La struttura interna dei database è personalizzabile per ogni divisione. In particolare la struttura scelta per le divisioni DVD e STB è la seguente:

Dvd
+Pkey: int
+recordtype: int
+author: String
+date: Date
+lastdate: Date
+category: String
+doctype: String
+ados: String
+family: String
+product: String
+salestype: String
+description: String
+keywords: String
+url: URL
+filename: String
+originalfilename: String
+internet: Boolean
+intranet: boolean
+intraprotect: Boolean
+validity: String
+owner: String
+group_owner: String
+oldner: String

Figura 4.3 struttura della tabella DVD nel database Div-Web

2.2 Enterprise Directory

L'Enterprise Directory [13] è il “*repository*” usato da altre applicazioni e servizi per fornire i servizi di Identificazione, Autenticazione, ed Autorizzazione (IAA) e controllo degli accessi, basati sulle informazioni memorizzate nel “*Directory*” attraverso il *protocollo LDAP*.

LDAP, Lightweight Directory Access, è lo standard utilizzato per fornire servizi di “Directory” distribuite su Internet. E' derivato dallo standard per i Directory X.500. La versione LDAP v2 è definita in RFC1777, mentre la versione LDAP v3 è definita in RFC 2252, quest'ultima è la versione utilizzata dall'Enterprise Directory .

Per spiegare nel miglior modo i servizi forniti dall'Enterprise Directory è necessario spiegare brevemente cosa sono Directory e Ldap [11] e [4].

2.2.1 Cos'è un Directory

Un Directory è un archivio di dati distribuito, non è un database relazionale generico (“general purpose”). Esso non è altro che un albero (ossia un grafo connesso ed aciclico) radicato.

Le caratteristiche principali di un directory sono :

- Visione statica dei dati;
- Un maggior numero di operazioni in lettura che in scrittura;
- Ottimizzazione per accesso in lettura;
- Aggiornamenti semplici (operazioni senza transazioni);

2.2.2 L'organizzazione dei dati in un Directory LDAP

La struttura dati fondamentale, almeno dal punto di vista astratto, è l'albero. Nello specifico, la struttura viene chiamata abitualmente **DIT**, ovvero *Directory Information Tree*.



Figura 4.4 DIT del STMicroelectronics

La radice dell'albero viene chiamata **base**, deve esistere ed essere unica per tutto il DIT.

Un directory contiene dei nodi chiamati **entry** od **oggetti**.

Un oggetto è una collezione di coppie chiave-valore chiamate **attributi**. Un attributo può contenere un solo valore, ed in questo caso è chiamato **single-valued attribute** oppure può ammettere più valori, e viene chiamato **multi-valued attribute**. Inoltre gli attributi possono essere obbligatori (**mandatory**) o opzionali (**optional**). Il numero ed il tipo degli attributi di un oggetto sono specificati da una o più classi (**ObjectClass**). Esistono tre tipi di classi:

- Astratte (Abstract).
- Strutturali (Structural).
- Ausiliarie (Auxiliary).

Le classi **astratte** vengono utilizzate solo per definire gli oggetti corrispondenti ai nodi interni di un albero e che non devono contenere dati significativi per l'utente.

Le classi **strutturali** definiscono la struttura fondamentale di un oggetto. Ogni oggetto deve essere istanza di una ed una sola classe strutturale.

Le classi **ausiliarie** definiscono attributi accessori per oggetti che sono già istanza di una classe strutturale, ed ogni oggetto ne può istanziare un numero a piacere, a patto che le classi accessorie siano compatibili con la classe strutturale.

La definizione delle classi in LDAP supporta l'ereditarietà: una classe può estendere una superclasse, ed in questo modo specializzarla. È inoltre possibile dichiarare come obbligatorio un attributo che nella superclasse era opzionale.

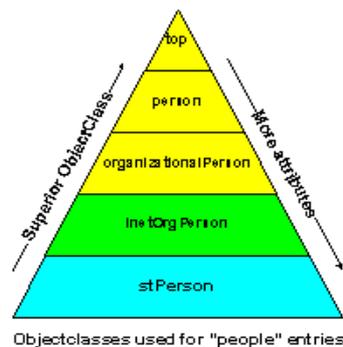


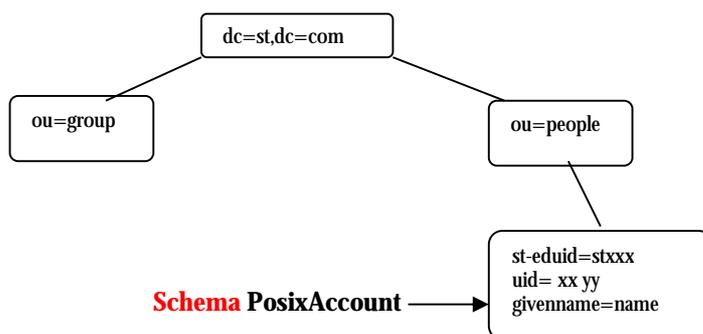
Figura 4.5 Esempio di classe in LDAP

Ogni classe ha un nome, ma questo serve solo per rendere facilmente identificabile la classe. Nella rappresentazione interna LDAP, ogni classe è identificata univocamente da un **OID** (Object Identifier), ovvero una stringa numerica che dev'essere **globalmente unica**. Proprio per quest'ultimo motivo, gli OID devono essere richiesti allo IANA (*Internet Assigned Number Authority*), che li fornisce gratuitamente, avendo cura che non ci siano due classi diverse con lo stesso OID (che non sono una caratteristica propria solo di LDAP o di X.500, ma vengono usati da moltissimi servizi diversi, quali SNMP).

La definizione di un insieme di classi e dei tipi degli attributi è detta **schema**.

Riassumendo, si può pensare ad una directory LDAP come ad un albero, in cui ad ogni nodo sono associate una o più coppie attributo-valore.

Ripercorrendo l'albero dalla foglia alla radice si ottiene il Distinguished Name (dn) che ne identifica univocamente una voce.



Nel caso in figura, per esempio, il dn è `steduid=xx,ou=people,dc=st,dc=com`.

2.2.3 Directory Distribuite

Uno degli aspetti più validi di X.500 e conseguentemente di LDAP è quello di essere stato progettato per consentire la creazione di **directory distribuite**. L'utilità a livello pratico è subito evidente: da una parte parti diverse di un organismo possono gestire in modo indipendente tra loro i dati che li riguardano, pur avendo sempre a disposizione l'intera base di dati, dall'altra si riduce il lavoro per i singoli *host* che eseguono il *directory server*: Il concetto alla base della distribuzione dei *directory* è quello di **partizione**.

Una partizione non è altro che un sottoinsieme dell'albero totale che sia anch'esso un albero, radicato in un cosiddetto **common ancestor**, ossia un antenato comune, che appartiene alla partizione stessa. Poiché un albero è una struttura dati ricorsiva, segue che si possono avere un numero arbitrario (almeno in teoria) di partizioni. Anche i database LDAP stessi possono riferirsi ad altri database per i livelli superiori o inferiori, diventando de facto una sorta di "partizione" di un insieme di database più grande.

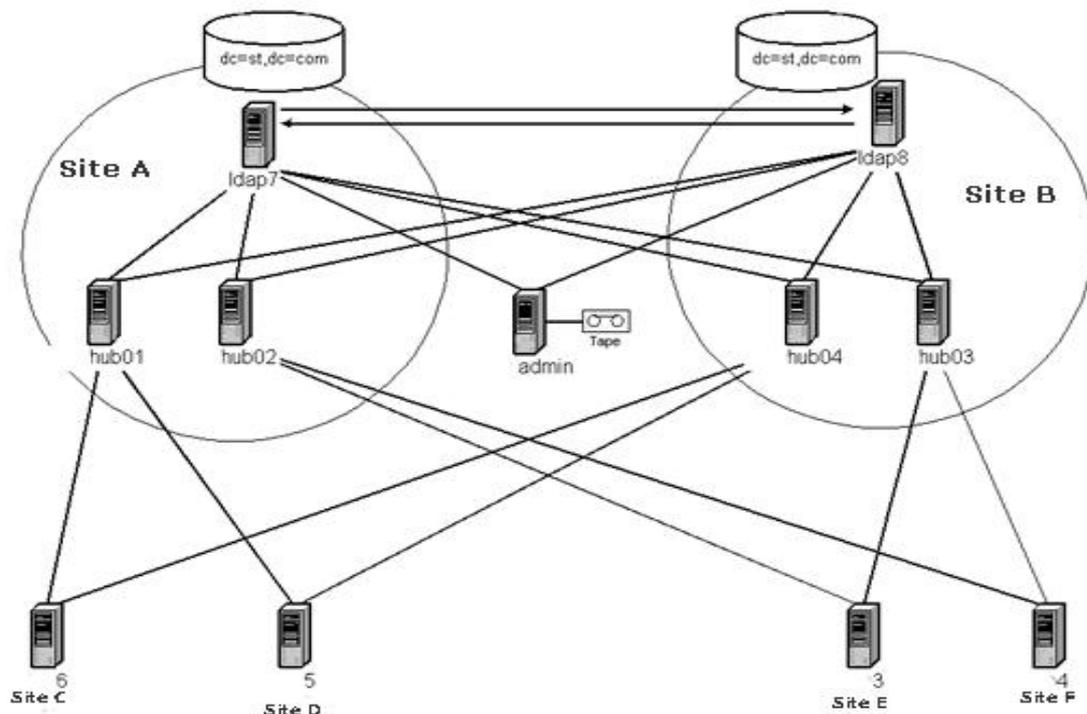


Figura 4.6. Struttura del directory distribuito all'interno del STMicronics

2.2.4 LDAP: Il protocollo

L'aspetto che differenzia realmente LDAP da X.500 è sicuramente il protocollo di comunicazione tra due server e tra client e server. Diamo ora una rapida panoramica degli aspetti salienti del protocollo, riferendoci sempre alla versione 3.

Per prima cosa, ogni operazione definita dal protocollo viene incapsulata in una busta (*envelope*), chiamata LDAPMessage, definita nel modo seguente:

```
LDAPMessage ::= SEQUENCE { messageID
MessageID, protocolOp CHOICE {
bindRequest
BindRequest, bindResponse BindResponse,
unbindRequest UnbindRequest,
searchRequest SearchRequest,
searchResEntry SearchResultEntry,
searchResDone SearchResultDone,
searchResRef SearchResultReference,
modifyRequest ModifyRequest,
modifyResponse ModifyResponse,
addRequest AddRequest,
addResponse AddResponse,
delRequest DelRequest,
delResponse DelResponse,
```

```

modDNRequest ModifyDNRequest,
modDNResponse ModifyDNResponse, compareRequest CompareRequest,
compareResponse CompareResponse,
abandonRequest AbandonRequest,
extendedReq ExtendedRequest,
extendedResp ExtendedResponse
}, controls [0] Controls OPTIONAL }
MessageID ::= INTEGER (0 .. maxInt)
maxInt INTEGER ::= 2147483647 -- (231 - 1) -

```

Se il server riceve una richiesta da parte di un client in cui il numero sequenziale SEQUENCE non può essere identificato, l'intero messaggio non è valido, e deve essere notificata la disconnessione.

Se il client riceve dal server una risposta con un SEQUENCE non valido, può decidere di scartarlo o di interrompere definitivamente la connessione.

Un altro aspetto saliente del protocollo è la codifica delle stringhe, che avviene seguendo lo standard ISO 10646, ed in particolare la codifica UTF-8, che preserva la notazione. Abbiamo già parlato dell'OID, definito dal tipo LDAPOID, che non è altro che una stringa formata da numeri e punti.

Per quanto riguarda i risultati di un query, questi sono inseriti in un messaggio LDAPResult, compresi gli errori, che per lo più assumono codici definiti nello standard X.509. Non c'è peraltro alcuna garanzia che un'operazione che restituisce un codice di successo sia completata del tutto, in altre parole non c'è necessariamente sincronia perfetta tra esecuzione e risultato, almeno in linea teorica.

LDAPResult ha una struttura molto semplice, definita come segue (chiaramente essa restituirà un solo resultCode alla volta):

```

LDAPResult ::= SEQUENCE { resultCode
ENUMERATED {
success (0), operationsError (1),
protocolError (2),
timeLimitExceeded (3),
sizeLimitExceeded (4),
compareFalse (5),
compareTrue (6),
authMethodNotSupported (7), strongAuthRequired (8),
-- 9 reserved --
referral (10), -- new

```

```

adminLimitExceeded (11), -- new unavailableCriticalExtension (12), -- new
confidentialityRequired (13), -- new
saslBindInProgress (14), -- new
noSuchAttribute (16),
undefinedAttributeType (17),
inappropriateMatching (18),
constraintViolation (19),
attributeOrValueExists (20),
invalidAttributeSyntax (21),
-- 22-31 unused - noSuchObject (
32), aliasProblem (33),
invalidDNyntax (34),
-- 35 reserved for undefined isLeaf -- aliasDereferencingProblem (36),
-- 37-47 unused -- inappropriateAuthentication (
48), invalidCredentials (49),
insufficientAccessRights (50),
busy (51), unavailable (52),
unwillingToPerform (53),
loopDetect (54),
-- 55-63 unused -- namingViolation (
64), objectClassViolation (65),
notAllowedOnNonLeaf (66),
notAllowedOnRDN (67),
entryAlreadyExists (68),
objectClassModsProhibited (69),
-- 70 reserved for CLDAP -- affectsMultipleDSAs (71), --
new
-- 72-79 unused --
other (80) },
-- 81-90 reserved for APIs --
matchedDN LDAPDN, errorMessage
LDAPString,
referral [3] Referral OPTIONAL }

```

Il protocollo è implementato tipicamente su TCP (Transmission Control Protocol). Ne esistono diverse implementazioni Open Source, tra cui le più famose sono: OpenLDAP ed Umich-Ldap (l'implementazione originaria dell'Università del Michigan).

2.2.5 La sicurezza in LDAP

LDAP v3 supporta diversi tipi di crittografia, tramite TLS (Transport Layer Security) derivato da SSL (Secure Sockets Layer), è possibile cifrare tutti i dati in entrata ed uscita dal server. Sono inoltre presenti sistemi d'autenticazione alternativi alle *password*

Dal punto di vista della struttura interna del database, un directory LDAP può contenere liste d'accesso (ACL) per ogni singolo attributo di un oggetto.

È possibile, anche, definire in modo molto semplice i diritti d'accesso ad un attributo, senza dover specificare in modo pedante gli utenti (ad esempio “self” sta ad identificare l'utente che si è identificato proprio tramite quell'oggetto).

2.2.6 A cosa serve LDAP?

- Autenticazione su interi sistemi o solo per alcuni servizi (quali pop ed imap, zone riservate di un server Web, ...).
- Distribuzione di configurazioni di sistema (per autofs, alias di posta, ...).
- Rubriche e schedari online.
- Archiviazione e ricerca di chiavi pubbliche (Keyring PGP).
- Archiviazione server-side di grosse zone DNS.
- Centralizza in un unico “repository” delle informazioni essenziali per gestioni di grandi aziende (ad esempio informazioni sui dipendenti).

In generale in qualsiasi situazione in cui si debbano gestire grosse quantità di coppie chiave-valore, in cui l'utilizzo di file di testo, perfino di database relazionali diventi inefficiente o macchinoso.

2.2.7 Enterprise Directory- Identificazione, Autenticazione, Autorizzazione

Il directory può essere usato per identificare ed autenticare l'utente e quindi controllare le informazioni accessibili da quest'ultimo (autorizzazione) secondo il metodo di **profiling** utilizzato.

Identificazione

Lo scopo di questo processo è verificare che l'utente è chi dice di essere. Questo si ottiene richiedendo login e password e quindi ricercando la coppia login/password nelle “entry” del directory.

Autenticazione

L'autenticazione effettuata usando l'Enterprise Directory avviene secondo il protocollo LDAP.

Profiling o identificazione di un tipo d'utente

Lo scopo di questo processo è controllare la quantità ed il tipo d'informazioni fornite ad un utente. Questo processo inizia dopo che il processo d'autenticazione è finito con successo.

L'uso del "profiling" permette di concedere l'accesso solo a determinati utenti. In questo modo, si può scegliere se pubblicare o non determinate informazioni. Il profiling può essere relativo ad attributi o a gruppi di persone.

Profiling effettuato controllando determinati attributi

Il profiling relativo ad attributi avviene controllando il valore contenuto in un determinato campo del directory. Questo tipo di profiling è preferibile perché l'accesso a determinati servizi varia dinamicamente al variare del valore in quel determinato attributo nel record di un utente.

Profiling effettuato controllando gruppi di persone

Esistono delle situazioni in cui il tipo di profiling citato sopra non è adatto, cioè quando non ci sono degli attributi adatti al profiling di un gruppo di persone o quando si vogliono utilizzare dei criteri d'autenticazione che non dipendono da un preciso valore contenuto in un attributo di un record, in questo caso si controlla se l'utente appartiene ad un gruppo di persone autorizzate.

In questi casi si utilizza il servizio fornito dall' Enterprise Directory chiamato "The Group Service", esso fornisce l'applicazione e gli strumenti per gestire questi gruppi d'utenti.

L'autenticazione da un punto di vista LDAP avviene tramite un'operazione speciale chiamata "bind". L'utente fornisce all'applicazione la sua login e password. Il risultato dell'autenticazione è fornito da un "LDAP bind".

Per eseguire un "LDAP bind" tramite una qualsiasi applicazione (implementata in un qualsiasi linguaggio di programmazione, PHP4, Perl, ecc.) essa deve disporre delle seguenti informazioni:

1. il DN dell'utente;
2. la password.

Per ottenere il DN di un utente quando si ha solo la login dell'utente, bisogna effettuare una ricerca nella Enterprise Directory attraverso una "LDAP search". Un "LDAP search" usa i seguenti parametri:

- Un LDAP base DN: che indica a quale livello si deve iniziare a cercare.
- Un LDAP scope: a quale livello di profondità nell'albero si deve effettuare la ricerca (per la ricerca dei dati riguardanti le persone, il livello è uno perché "people" è un ramo piatto).
- Un LDAP search filter: indica cosa cercare.
- Una lista di attributi LDAP da prelevare: cioè quale coppia attributo/valore si vuole ritornato da una ricerca.

Un tipico scenario per l'autenticazione si può vedere in figura.

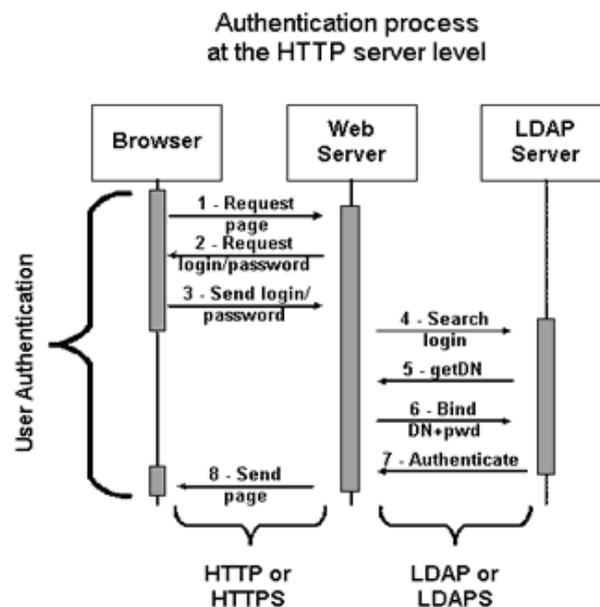


Figura 4.7. Tipico scenario di autenticazione

1. Si apre una connessione con un server LDAP.
2. Si controlla se la connessione è avvenuta con successo.
3. Si preleva il DN dell'utente attraverso una LDAP search usando la login dell'utente come stringa.
4. Si esegue un LDAP "bind" usando il DN dell'utente e la password.
5. Si controlla se la "bind" è stata effettuata con successo (l'utente è stato autenticato) o meno (l'utente ha fornito una password errata).

6. Si effettua una disconnessione dal server LDAP.

Quando il processo di autenticazione dell'utente è completato, l'identità dell'utente è nota. Il prossimo passo è l'autorizzazione dell'utente.

Autorizzazione

Esistono due modi differenti per autorizzare un utente attraverso l'Enterprise Directory:

- Controllando che l'utente ha un valore ben noto in uno o più attributi del suo record (Profiling effettuato controllando determinati attributi).
- Controllando che l'utente appartenga ad un gruppo indicato nell'Enterprise Directory attraverso l'applicazione "The Group Service" (Profiling effettuato controllando gruppi di persone).

2.3 The Group Service

The Group Service è nato dall'esigenza di creare e mantenere informazioni riguardanti gruppi di persone o "distribution list". Attraverso questo servizio, altre applicazioni potranno usare gruppi di persone per diversi scopi (controllo degli accessi, profiling) e le liste di distribuzione per un invio automatico di e-mail.

Per la memorizzazione e la gestione dei gruppi di persone e delle liste di distribuzione si utilizza l'Enterprise Directory.

The Group Service è un'applicazione Web, utilizzabile solo da dipendenti ST (autenticati), per la creazione e manutenzione dei gruppi. Tramite essa si possono effettuare tre azioni principali:

1. Creare e cancellare gruppi.
2. Gestire il contenuto dei gruppi.
3. Visualizzare i gruppi.

In breve, tramite quest'applicazione è possibile inserire nell'Enterprise Directory gli oggetti necessari per la creazione e l'utilizzo di questi gruppi, oggetti che verranno usati da altre applicazioni che necessitano di queste informazioni.

Un gruppo contiene le seguenti informazioni:

- **Name:** valore singolo della entry della directory, composta da una prima parte, 5 – 8 caratteri, scelta in una lista di parole chiave, il carattere “underscore”, ed una stringa composta da 1 a 18 caratteri.
- **Description:** descrizione del gruppo.
- **Owner:** DN del utente ST che ha richiesto la creazione del gruppo. (Un utente può avere al massimo 50 gruppi).
- **Costcenter:** centro di costo della persona che richiede la creazione del gruppo.
- **UniqueMember:** lista (attributo multivalore) dei membri del gruppo. Per un gruppo di persone esso contiene il DN delle persone, mentre per le liste di distribuzione esso conterrà gli indirizzi e-mail (formato concorde a RFC822). Al massimo possono essere presenti 720 valori.
- **Administrators:** Contiene il DN delle persone che sono autorizzate per la gestione delle liste dei membri. Questo campo può contenere al più 10 valori. Un amministratore non può gestire più di 50 gruppi.
- **Type:** sono supportate 3 tipi di visibilità:
 1. **Pubblica :** tutti possono leggere il contenuto della lista dei membri del gruppo (dopo essere stati autenticati).
 2. **Privata:** Solo i membri della lista possono leggerne il contenuto.
 3. **Hidden:** solo gli amministratori di un gruppo possono leggere il contenuto della lista.

IL CASO DI STUDIO IP EXCHANGE

Nell'attività di sviluppo di SoC⁷ la progettazione e l'integrazione di IP⁸ rappresentano le fasi più onerose tanto che spesso tali attività vengono eseguite presso divisioni indipendenti tra loro. Al fine di assicurarne l'integrabilità, nasce l'esigenza di definire una metodologia che ne formalizzi lo scambio e garantisca l'intesa tra chi sviluppa e chi integra.

1 Introduzione

La divisione CMGDesign del gruppo CMG della STMicroelectronics, da tempo lavora alla modellazione del processo di scambio degli IP, approdando ad una procedura formale [3] che ne descrive il flusso, ed un prototipo d'uso su Web, che ne facilita l'utilizzo. Visto l'ampio consenso riscosso dal prototipo, la CMGDesign mi ha chiesto la progettazione e realizzazione dell'applicazione "IP Exchange", del database di supporto, e di alcuni moduli specifici che integrano altre applicazioni già in uso in STMicroelectronics.

2 Requisiti Funzionali

Il processo inizia tramite una specifica azione del richiedente, il quale spedisce una richiesta al fornitore.

Nel caso in cui l'IP richiesto non esiste, oppure esiste ma il cliente richiede qualche modifica, il fornitore deve valutare il costo dello sviluppo o delle modifiche relative, e quindi deve rispondere al cliente con una proposta, contenente i dettagli sul lavoro da realizzare e le modalità di consegna.

A questo punto il richiedente può accettare la proposta, per cui le parti concordano la fase di sviluppo, oppure può chiedere di negoziare alcuni dettagli, o in casi estremi può rifiutare la proposta. Se il cliente richiede ulteriori modifiche rispetto a quelle già presenti nella richiesta (ECR, Enhancement Change Request), e comunque a sviluppo già concordato, il fornitore

⁷ System On Chip, sistemi complessi altamente integrati su un'unica porzione di silicio.

⁸ Intellectual Property, blocco funzionale descrivibile a vari livelli di astrazione.

valuta il costo incrementale e quindi lo rende noto al richiedente, con un'ulteriore proposta indipendente da quella già in atto.

Se invece l'ECR viene inviato prima che il fornitore abbia emesso la proposta, allora la vecchia richiesta e lo ECR verranno esaminati insieme e la proposta quindi sarà unica e cumulativa.

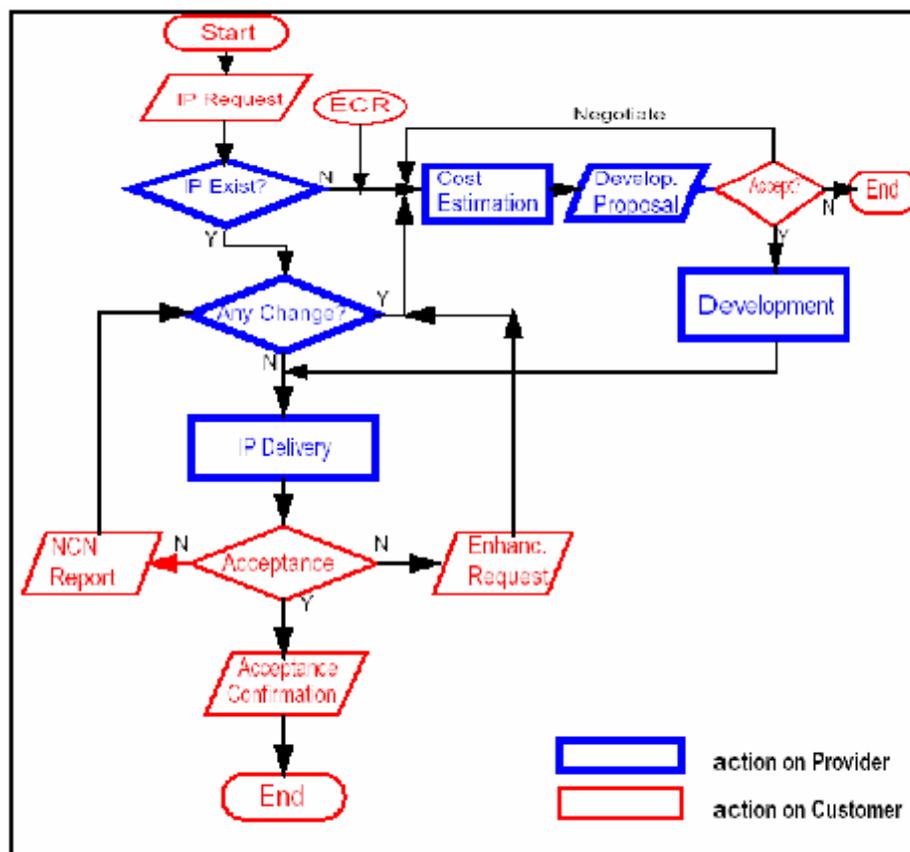


Figura 5.1. flusso richiesta/consegna IP

Se il cliente richiede un IP esistente senza nessuna modifica, il fornitore consegnerà il pacchetto dell'IP corrispondente ed attenderà il questionario dal cliente entro un periodo di tempo prefissato. Quando il cliente invierà questo documento al fornitore la procedura è completata, altrimenti il cliente segnalerà i problemi che ha incontrato.

Se i problemi segnalati dal cliente sono facilmente risolvibili (NCN Report, No Conformance), il fornitore può effettuare delle piccole modifiche e quindi rilasciare una nuova versione. Se invece, la risoluzione di quest'ultimi richiede delle attività di progettazione consistenti (Enhancement Request), il cliente può decidere di abortire la richiesta, che quindi sarà chiusa con fallimento, altrimenti può richiedere che queste modifiche siano effettuate, e quindi aspettare la risposta del fornitore, così come avviene per la proposta.

L'intera procedura è schematizzata nel diagramma di flusso in figura 5.1.

2.1 Procedure per lo scambio degli IP da automatizzare

In questo paragrafo, sono analizzate in dettaglio le procedure descritte in figura 5.1, per le quali è stata richiesta l'automazione.

Le procedure ECR, NCN Report ed Enhancement Request saranno sviluppate in seguito.

2.1.1 Procedura IP Request

Questa è la procedura tramite la quale il cliente inizia tutto il processo. Il cliente dovrebbe visionare il catalogo e quindi riempire il modulo per la richiesta disponibile On line.

Quando il fornitore riceve la richiesta, l'analizza per vedere se l'IP richiesto è già esistente o meno. Se l'IP è già esistente il fornitore può velocemente consegnare al cliente l'IP richiesto. Altrimenti, invierà una notifica al cliente indicando una data approssimativa entro la quale s'impegnerà ad inviare una proposta. Se per una qualsiasi ragione, il fornitore non può soddisfare la richiesta lo renderà noto al cliente e la negoziazione potrà ritenersi conclusa.

2.1.2 Procedura IP proposal

Nel caso in cui è stato richiesto un nuovo IP, oppure uno già esistente ma con delle variazioni il fornitore valuterà il costo d'implementazione in termini di uomo/mese, e invierà una proposta al cliente, il quale la potrà accettare, negoziare o rifiutare.

Se la proposta sarà rifiutata, la richiesta sarà sospesa o eventualmente chiusa. Altrimenti, se cliente vorrà negoziare qualche dettaglio, potrà segnalarlo direttamente al fornitore il quale

provvederà a stilare un'altra proposta di sviluppo. Questo processo andrà avanti finché il richiedente accetterà o rifiuterà formalmente la proposta.

2.1.3 Procedura IP acceptance

Successivamente alla consegna dell'ultimo pacchetto relativo all'IP richiesto, il fornitore aspetta un ragionevole lasso di tempo, trascorso il quale invierà al cliente un questionario per la valutazione del servizio usufruito. La restituzione del questionario compilato, sarà l'evento di chiusura dell'intera transazione.

3 Requisiti non Funzionali

Per la struttura dell'hardware e del software utilizzati si veda il capitolo 4.

4 Funzionalità che IP EXCHANGE offre agli utenti

In breve, le funzionalità offerte sono:

1. Visione del catalogo.
2. Autenticazione dell'utente tramite l'Enterprise Directory.
3. Richiesta di IP via Web da un dipendente ST autenticato.
4. Gestione e processamento della richiesta, riservato agli utenti registrati come "Provider".
5. La richiesta può essere:
 - Rifiutata.
 - Accettata, quindi si stabilisce una data approssimativa intorno la quale il fornitore s'impegnerà ad inviare una Proposta.
 - Sospesa.
 - Avanzata direttamente ad uno stato di consegna, in quanto è stato richiesto un IP già esistente.
6. Se la richiesta è stata accettata dal fornitore, lo stesso provvede a stilare una proposta.
7. L'utente che ha inviato la Richiesta può accettare, rifiutare o negoziare la proposta.
8. Gestione degli IP:
 - Avanzamento lo stato della richiesta/proposta.
 - Notifica dell'avvenuto rilascio.
 - Invio del modulo per il questionario sul servizio.
 - Visione della richieste.
 - Visione la proposte.

- Invio del questionario.

5 Casi d'Uso

5.1 Attori

Gli attori nella descrizione dei requisiti sono: Customer, Provider, Provider Manager, Administrator.

Actor: Customer.
Description: Un Customer è un dipendente ST, autenticato, che manda una richiesta di IP.
Use Case Authentication. Authentication Customer. Send IP Request. Accept/Refuse Proposal. Fill and send Questionnaire.

Actor: Provider.
Description: Un Provider è un dipendente ST appartenente ad una divisione che fornisce IP, il quale fornisce uno o più tipi specifici d'IP (ad esempio audio, video, ecc).
Use Case Manage IP. Check index. Active request. Authentication. Authentication Provider. View Request. Send Proposal. Change Status of IP request. Send questionnaire. Delete IP. Delete IP Request. Delete Proposal. Delete Questionnaire.

Actor: Provider Manager.
Description: Un Provider Manager è un dipendente ST, responsabile di tutta la gestione degli IP di una determinata divisione. Esso può gestire tutti i "tipi" di IP di una divisione specifica ed, inoltre, è informato delle transazioni. Ha le stesse funzionalità del provider, ma estese a tutti gli IP della divisione di appartenenza, seppur gestiti da specifico "Provider" precedentemente registrato e comunque appartenente alla stessa divisione.

<p>Use Case Manage IP. Check index. Active request. Authentication. Authentication Provider. View Request. Send Proposal. Change Status of IP request. Send questionnaire. Delete IP. Delete IP Request. Delete Proposal. Delete Questionnaire.</p>

<p>Actor: Administrator</p>
<p>Description: Un Administrator è una persona a cui è stato affidato il compito di aggiornare e mantenere tutti i dati necessari per IP Exchange.</p>
<p>Use Case Authentication. Authentication Administrator. Manage People. Insert Provider. Modify People . Delete People. Insert Provider Manager. Insert new People. Manage Information about Database Management System. Insert new Database Management System. Modify DMS. Delete DMS. Manage Information about Technology. Insert new Technology. Modify Technology. Delete Technology. Manage IP Division. Insert new IP Division. Modify IP Division. Delete Division. Manage IP Type. Insert new IP Type. Modify IP Type. Delete IP Type.</p>

5.2 Casi d'uso in dettaglio

In questa sezione verranno analizzati i casi d'uso. L'analisi inizia dai casi d'uso relativi alla figura 5.2

Use case name: "Authentication".

Actors: Customer, Provider, Provider Manager, Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'utente clicca sul link "How To Request an IP" o sul link "IP Management".

Post-condition: l'utente è stato riconosciuto dal sistema.

Use Case that "extends" this: "Authenticate Provider", "Authenticate Customer", "Authenticate Administrator".

Flow of events:

1. Il caso d'uso inizia quando è visualizzata la finestra che consente l'inserimento di Login e Password.
2. In questa finestra ci sono due bottoni: "authenticate" e "cancel".
3. Se l'utente:
 - a) Seleziona il bottone "cancel", non verrà autenticato, appare quindi un messaggio d'errore.
 - b) Seleziona il bottone "ok":
 1. Il sistema si collega al server ldap.
 2. Ricerca un record che contenga la login fornita dall'utente, nella "Enterprise Directory".
 3. Verifica se la password inserita coincide con quella memorizzata nel record appena prelevato.
4. fine del caso d'uso.

Secondary scenario:

- A. 3.b) 1. Il server ldap non è disponibile.
 2. L'utente non si potrà autenticare.
 3. Verrà visualizzato un messaggio d'errore.
- B. 3.b) 1. La login fornita è sbagliata
 2. L'utente non sarà autenticato, potrà riprovare di nuovo ad inserire una login giusta.
- C. 3.b) 1. La Password fornita è sbagliata.
 2. L'utente non sarà autenticato, potrà riprovare di nuovo ad autenticarsi.

Use case name: “Authentication Customer”.

Actors: Customer.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'utente che ha selezionato il link “How To Request an IP” è stato già autenticato dal caso d'uso “Authentication”.

Post-condition: I dati del Customer sono stati prelevati dall'Enterprise Directory.

Flow of events:

1. Vengono recuperati i dati del Customer, necessari per effettuare la richiesta: nome, cognome, e-mail, telefono, gruppo, divisione e sede ST d'appartenenza.
2. Il Customer è adesso autenticato può effettuare la sua richiesta.

Use case name: “Authentication Provider”.

Actors: Provider, Provider Manager.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'utente che ha selezionato il link “IP Management” è stato già autenticato dal caso d'uso “Authentication”.

Post-condition: Il provider è stato riconosciuto dal sistema, gli sarà consentito l'accesso all'area IP management.

Flow of events:

1. L'utente autenticato appartiene all'insieme dei provider.
2. Si verifica se:
 - a) Il provider è manager di una divisione.
 - ⇒ Potrà accedere all'area IP Management.
 - ⇒ Gli sarà concesso l'accesso a tutti i dati riguardanti, la divisione cui fa capo.
 - b) Il provider non è manager.
 - ⇒ Potrà accedere all'area IP Management.
 - ⇒ Gli sarà concesso l'accesso a tutti i dati riguardanti, solo le richieste, proposte, ecc. per le quali lui risulta provider.
3. fine caso d'uso.

Secondary scenario:

1. L'utente autenticato non appartiene all'insieme dei provider.
2. Non potrà accedere all'area IP Management.
3. Fine caso d'uso.

Use case name: “Authentication Administrator”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’utente che ha selezionato il link “IP Administration” è stato già autenticato dal caso d’uso “Authentication”.

Post-condition: L’administratore è stato riconosciuto dal sistema, gli sarà consentito l’accesso all’area IP administrator.

Flow of events:

1. L’utente autenticato appartiene all’insieme degli administrator.
2. Potrà accedere all’area IP Administration.

Secondary scenario:

1. L’utente autenticato non appartiene all’insieme degli administrator.
2. Non potrà accedere all’area IP Administration.

Use case name: “Send IP Request”.

Actors: Customer.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’utente è autenticato, i suoi dati sono disponibili per la richiesta. Il form per acquisire la richiesta è visualizzato, i campi relativi all’utente sono riempiti in automatico.

Post-condition: La richiesta e i dati del customer sono stati registrati.

Use case “included”: “Send Notify”

Flow of events:

1. Il caso d’uso inizia quando l’utente compila il form con tutti i dati relativi alla sua richiesta.
2. L’utente dovrà obbligatoriamente indicare l’ip_name e l’ip_functionality, ma soprattutto dovrà scegliere l’IP Division e l’IP Type (dati necessari per individuare il provider a cui inviare la richiesta).
3. Quando l’utente avrà inserito i dati e spedito la richiesta, verranno prelevati tutti i dati dal form.
4. Si controllerà che i dati obbligatori siano stati inseriti.
5. Inoltre si controllerà che i campi che indicano le scadenze per la consegna delle versioni parziali o finali, siano degli interi compresi fra 1 e 52 (settimane lavorative in un anno).
6. Si ricerca il provider che fornisce il tipo di IP richiesto per la divisione scelta.
7. Se:
 - a) Non vi è associato un provider al tipo scelto, si considera come provider il “provider manager” della divisione in questione.
 - b) Altrimenti il provider è quello associato al tipo.

8. Il customer seleziona il bottone “send request”, la richiesta è memorizzata. Assumerà in automatico lo stato “new”.
 9. Il controllo passa al caso d’uso “Send notify”, al quale vengono passati i seguenti parametri:
 - a) From=indirizzo e-mail del provider.
 - b) To=indirizzo e-mail del customer.
 - c) CC= indirizzo e-mail del provider manager.
 - e) Subject="New IP request: “ip_name”.
 - f) Body="Dear Customer,
 many thanks for Your contact.
 Your request has been logged and You can check the status on “name of page”.
 Should You require any clarification, please do not hesitate to contact us.
 Regards, IP group".
 10. fine del caso d’uso.
- Secondary scenario:**
- A.
 2. L’IP name o l’IP functionality o IP division o IP type non sono stati inseriti.
 3. La richiesta non potrà essere inoltrata, un messaggio d’errore ne segnalerà il motivo.
 4. Fine del caso d’uso.
 - B.
 5. i campi che indicano le scadenze per la consegna delle release parziali o finali, non sono degli interi compresi fra 1 e 52.
 6. la richiesta non potrà essere inoltrata, un messaggio d’errore ne segnalerà il motivo.

Use case name: “Manage IP”

Actors: Provider, Provider Manager.

Priority: 1

Status: Requirements capture

Pre-condition: Nella Home Page è stato selezionato il link ad “IP Management”, Il provider è autenticato.

Use case “included”: “Active Request”, “Check Index”, “Send form for questionnaire”, “Delete IP”.

Flow of events:

1. La pagina di IP_management è adesso visualizzata.
2. Da questa pagina l’utente può effettuare diverse operazioni.
3. Sono disponibili le seguenti opzioni:
 - a) “Check Active Request”, se selezionato il controllo passa al caso d’uso “Active Request”.
 - b) “Check Index”, se selezionato il controllo passa al caso d’uso “Check Index”.

- c) “Send form for questionnaire”, se selezionato il controllo passa al caso d’uso “Send form for questionnaire”;
- d) “Delete IP”, se selezionato il controllo passa al caso d’uso “Delete IP”.

Use case name: “Accept / Refuse Proposal”.

Actors: Customer.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il customer riceve la notifica tramite la quale lo si avverte che il provider ha inoltrato la proposta alla sua richiesta.

Post-condition: Il customer ha accettato, rifiutato o negoziato la proposta.

Use case “included”: “Send Notify”.

Flow of events:

1. Il customer seleziona l’URL presente nella notifica.
2. Tutti i dati della proposta sono adesso visualizzati, ad essi sarà aggiunta una zona dedicata al customer, che gli permetterà di accettare, rifiutare o negoziare la proposta.
3. Il customer analizza la proposta.
4. Il customer può :
 - a) Accettare la richiesta e aggiungere degli eventuali commenti.
 - b) Rifiutare la richiesta e aggiungere degli eventuali commenti.
5. Il controllo passa al caso d’uso “Send notify”. Al quale vengono passati i seguenti parametri:
 - a) From=indirizzo e-mail del customer.
 - b) To=indirizzo e-mail del provider.
 - c) CC= indirizzo e-mail del provider manager.
 - d) Subject=”Accept/RefuseProposal “ip_name”“.
 - e) Body=”Dear “name of provider”,
your proposal for the “ip_name” to “name of customer” has been “accepted or refused” with the following comments: “notes””.

Use case name: “Fill and send questionnaire”.

Actors: Customer.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il customer riceve la notifica tramite la quale lo si avverte che il provider ha inoltrato il questionario.

Post-condition:.

Use case “included”: “Send Notify”.

Flow of events:

1. Il customer seleziona l'URL presente nella notifica.
2. Verrà visualizzato un form attraverso il quale il customer può rispondere alle domande del questionario.
1. Quando il customer ha finito la compilazione del questionario, clicca sul bottone “send questionnaire”.
2. Il controllo passa al caso d'uso “Send notify”, al quale vengono passati i seguenti parametri:
 - a) From=indirizzo e-mail del customer.
 - b) To=indirizzo e-mail del provider.
 - c) Subject=Customer Satisfaction Survey for: "ip_name".
 - d) Body=”The questionnaire for the request N."id" ("ip_name") is on “name of page”.

Use case name: “Send Notify”.

Actors: Customer, Provider, Provider Manager.

Priority: 1.

Status: Requirements capture.

Pre-condition: From, To, cc,subject, body sono stati passati da un altro caso d'uso.

Post-condition: La notifica è stata inviata.

Flow of events:

1. From, To, cc,subject, body vengono formattati per creare una e-mail.
2. L'e-mail viene inviata.
3. E-mail spedita senza problemi, il caso d'uso finisce qui.

Secondary scenario:

3. Spedizione dell' e-mail avvenuta senza successo.
4. Viene visualizzato mandato un messaggio d'errore che indica il tipo di problema riscontrato.

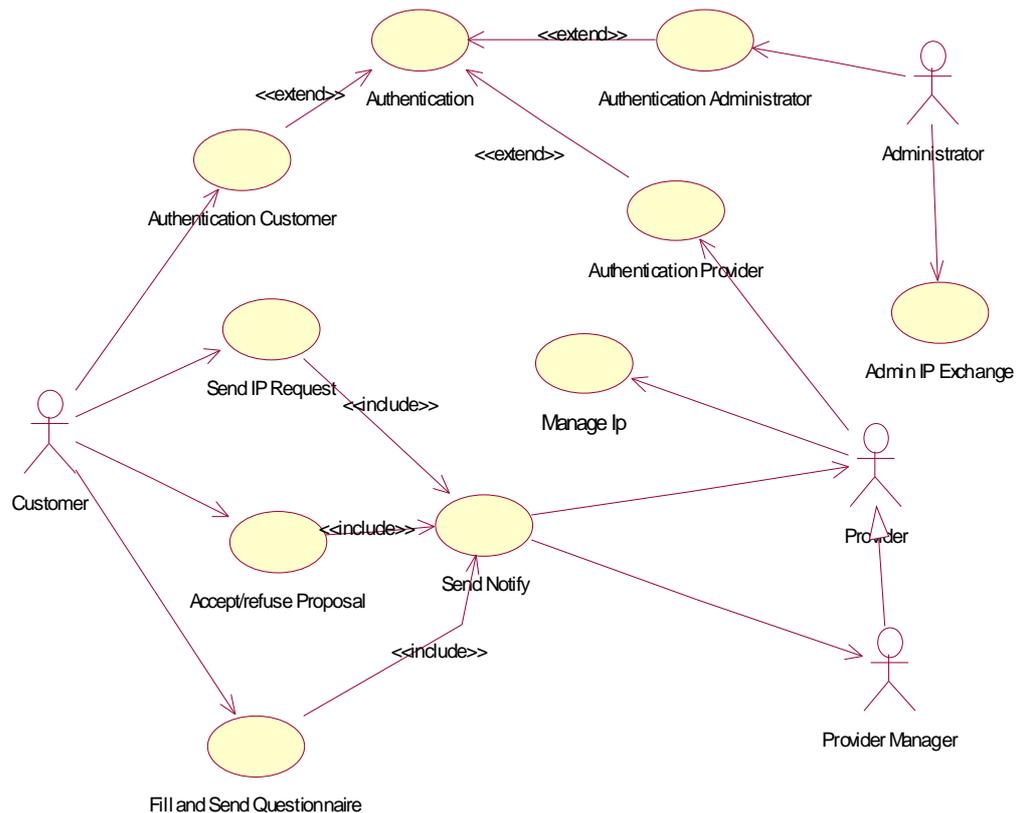


Figura 5.2 Diagramma dei casi d'uso (Vista Principale)

L'analisi continua con i casi d'uso relativi alla figura 5.3. Essi rappresentano il raffinamento del caso d'uso "Manage IP".

<p>Use case name: "View Request". Actors: Provider, Provider Manager. Priority: 1. Status: Requirements capture. Pre-condition: Nella pagina Active Request viene segnalata una nuova richiesta. Lo stato della richiesta è "new". Post-condition: La richiesta ha cambiato stato. Use case "included": "Send Notify". Flow of events:</p> <ol style="list-style-type: none"> 1. Il provider dopo aver ricevuto la notifica che segnala una nuova richiesta si collega alla pagina "active request", nella quale fra le altre richieste ci sarà la richiesta in

- questione, avente come stato “new”.
2. Selezionando il link corrispondente verranno visualizzati tutti i dati inseriti dal customer nella sua richiesta.
 3. Dopo una breve analisi della richiesta il provider potrà:
 - a) Accettare la richiesta, ed indicare una data approssimativa intorno alla quale s’impegnerà ad inviare una Proposta al customer, ed aggiungere delle eventuali note da inviare al Customer. Lo stato della richiesta verrà modificato in “logged”.
 - b) Sospendere la richiesta indicandone il motivo. Lo stato della richiesta verrà modificato in “suspended”.
 - c) Rifiutare la richiesta indicandone il motivo. Lo stato della richiesta verrà modificato in “closed”.
 - d) Avanzare la richiesta ad uno stato di consegna perché l’IP richiesto è già esistente. Lo stato della richiesta verrà modificato in “ongoing”.
 4. Il controllo passa al caso d’uso “Send notify”, per informare il customer sulle variazioni di stato della sua richiesta.
 5. Vengono passati i seguenti parametri:
 - a) From=indirizzo e-mail del provider.
 - b) To=indirizzo e-mail del customer.
 - c) Subject e Body variano secondo la scelta fatta dal provider.

Use case name: “Active Request”

Actors: Provider, Provider Manager

Priority: 1

Status: Requirements capture

Pre-condition: Nella pagina IP_management è stato selezionato il link ad “Check Active Request”. Il provider è autenticato.

Use case “included”: “View Request”, “Send Proposal”, “Change Status of IP”

Use Case that “extends” this: “Check Index”.

Flow of events:

1. La Pagina Active request è adesso visualizzata.
2. Se il Provider è un “Provider Manager” allora potrà accedere a tutte le richieste appartenenti alla divisione di appartenenza.
3. Se il Provider non è Manager potrà accedere solo a quelle richieste per le quali lui risulta “provider”.
4. In questa Pagina saranno visibili tutte le richieste non ancora completate.

5. Da questa Pagina si potrà vedere un prospetto riassuntivo della richiesta così com'è stata inviata dal customer. Se la richiesta si trova nello stato "new" il controllo passa al caso d'uso "View Request".
 6. Se lo stato della richiesta è "logged" sarà possibile mandare la proposta. Il controllo passa al caso d'uso "Send Proposal".
 7. Se invece una proposta già è stata inviata dal provider, da questa pagina si potrà vedere un prospetto riassuntivo di essa.
- Se invece il provider intende avanzare lo stato della richiesta il controllo passa al caso d'uso "Change Status of IP".

- Use case name:** "Check Index"
Actors: Provider, Provider Manager
Priority: 1
Status: Requirements capture
Pre-condition: Nella pagina IP_management è stato selezionato il link ad "Check Index". Il provider è autenticato.
Use case "included": "View Request", "Send Proposal", "Change Status of IP"
Flow of events:
1. Sono visibili tutte le richieste anche quelle concluse.
 2. Tutte le funzionalità presenti in "Active Request" sono disponibili anche qui.
 3. Se la Transizione è conclusa è possibile anche visualizzare un prospetto riassuntivo del questionario.

- Use case name:** "Send Proposal".
Actors: Provider, Provider Manager.
Priority: 1.
Status: Requirements capture.
Pre-condition: Lo stato della richiesta è "logged" o "suspended" o "proposed".
Post-condition: La proposta è stata memorizzata. Lo stato della richiesta passa a "proposed".
Use case "included": "Send Notify".
Flow of events:
1. Il provider seleziona il link "new proposal" dalla pagina "active request".
 2. Viene visualizzato il form tramite il quale sarà possibile inoltrare la proposta.
 3. I campi corrispondenti alla richiesta saranno in automaticamente caricati nel modulo di proposta. Il provider potrà modificare gli elementi, che non sono soddisfacenti.
 4. Il provider seleziona il bottone "send proposal", la proposta è memorizzata. Lo stato

della richiesta passa automaticamente in “proposed”.

5. Il controllo passa al caso d'uso “Send notify”. Al quale vengono passati i seguenti parametri:

- a) From=indirizzo e-mail del provider.
- b) To=indirizzo e-mail del customer.
- c) Subject=Proposal for "ip_name." (req_id: "id") .
- d) Body=”Dear Customer,
Your request has been processed and our proposal is at: ”name of page”.
Please note that as next step, we expect You to reply with an acceptance or refusal and meanwhile Your request will be put on hold.”.

Use case name: “Send questionnaire”.

Actors: Provider, Provider Manager.

Priority: 1.

Status: Requirements capture.

Pre-condition: Lo stato di richiesta-proposta è “completed”, il provider seleziona il link “send questionnaire” dal menu dell’ area “IP management” .

Post-condition: Il questionario è stato inviato al customer.

Use case “included”: “Send Notify”.

Nota:Quando richiesta e proposta si trovano nello stato “completed”, il provider desidera avere un resoconto della qualità del prodotto e del servizio offerto, quindi manda un questionario di valutazione al customer .

Flow of events:

1. Il provider seleziona il link “send questionnaire” e compila il modulo con il nome del customer, l’indirizzo e-mail del customer e del provider, e l’identificativo dell’ IP per cui si chiede la valutazione.

2. Quando il provider clicca il bottone *send*, il controllo passa al caso d’uso

“Send notify”. Al quale vengono passati i seguenti parametri:

- a) From=indirizzo e-mail del provider.
- b) To=indirizzo e-mail del customer.
- c) Subject=Customer Satisfaction Survey for: "ip_name".
- d) Body=”Dear “name of Customer”,
as consequence of our recent contact about "ip_name", we are very much interested in Your perception of our service quality and hence, we would ask You to fill our questionnaire.
It is fundamental to value actual service and promote future improvements.
Please note that Your feedback is very important to us

and we might remind You its return if not received
by "date";
To fill the questionnaire just click on the following
address "URL".
Should You require any assistance, do not hesitate to
contact me.
Thanks in advance for your co-operation
Regards, IP Group".

Use case name: "Change status of IP Request".

Actors: Provider, Provider Manager.

Priority: 1.

Status: Requirements capture.

Pre-condition: E' stato selezionato il link "change status" per un IP specifico.

Post-condition: Lo stato della richiesta_proposta è stato cambiato.

Flow of events:

1. Se lo stato iniziale è:

a) "new", lo stato può essere cambiato in:

1. "suspended".

b) "logged", lo stato può essere cambiato in:

1. "suspended".

2. "closed".

c) "proposed", lo stato può essere cambiato in:

1. "ongoing".

2. "completed".

3. "suspended".

4. "proposed".

d) "suspended", lo stato può essere cambiato in:

1. "logged".

2. "ongoing".

3. "completed".

4. "closed".

5. "proposed".

e) "completed", lo stato può essere cambiato in:

1. "suspended".

2. "closed".

f) “ongoing”, lo stato può essere cambiato in:

1. “completed”.
2. “suspended”.
2. Si scelga lo stato nuovo.
3. selezionando il bottone “change” lo stato verrà cambiato.

Secondary scenario:

3. selezionando il bottone “cancel” tutto rimarrà invariato.

Use case name: “Delete IP”.

Actors: Provider, Provider Manager.

Priority: 1.

Status: Requirements capture.

Pre-condition: E’ stato selezionato “delete” dal menu dell’area “IP Management”.

Post-condition: L’IP selezionato non esiste più.

Use case “included”: “Delete IP request”, “Delete Proposal”, “Delete Questionnaire”.

Flow of events:

1. La lista degli IP appartenenti al provider che effettua l'operazione è adesso visualizzata.
2. Si può scegliere l'identificativo di IP da cancellare, ed in particolare cosa.
3. Se si sceglie di cancellare:
 - a. La richiesta, allora entra in gioco il caso d’uso “Delete IP request”.
 - b. La proposta, allora entra in gioco il caso d’uso “Delete Proposal”.
 - c. Il questionario, allora entra in gioco il caso d’uso “Delete Questionnaire”.

Use case name: “Delete IP Request”.

Actors: Provider, Provider Manager.

Priority: 1.

Status: Requirements capture.

Pre-condition:E’ stato selezionato l’IP da cancellare.

Post-condition:L’IP selezionato non esiste più.

Use case “included”: “Delete Proposal”, “Delete Questionnaire”.

Flow of events:

1. Si cancella la richiesta con identificativo passato dal caso d’uso “Delete IP”.
2. Se esiste la proposta relativa alla richiesta cancellata si cancella anch’essa e quindi entra in gioco “Delete Proposal”.
3. Se esiste il questionario relativo alla richiesta cancellata si cancella anch’esso e quindi entra in gioco “Delete Questionnaire”.

Use case name: “Delete Proposal”.
Actors: Provider, Provider Manager.
Priority: 1.
Status: Requirements capture.
Pre-condition: E’ stato selezionato l’IP da cancellare.
Post-condition: La proposta dell’IP selezionato non esiste più.
Flow of events:
1. Si cancella la proposta con identificativo passato dal caso d’uso chiamante.

Use case name: “Delete Questionnaire”.
Actors: Provider, Provider Manager.
Priority: 1.
Status: Requirements capture.
Pre-condition: E’ stato selezionato l’IP da cancellare.
Post-condition: Il Questionario dell’IP selezionato non esiste più.
Flow of events:
1. Si cancella il questionario con identificativo passato dal caso d’uso chiamante.

2. Su ogni DMS esistente si possono effettuare due operazioni, a scelta : modifica e cancellazione.
3. Si può eventualmente inserire un nuovo DMS.
4. Se viene scelta l'azione:
 - a. "insert new", il controllo passa al caso d'uso "Inserito New DMS";
 - b. "modify", in corrispondenza di un DMS già esistente, il controllo passa al caso d'uso "Modify DMS ", al quale viene passato anche il DMS da modificare;
 - c. "delete", in corrispondenza di un DMS già esistente, il controllo passa al caso d'uso "Delete DMS ", al quale viene passato anche il DMS da cancellare.

Use case name: "Insert DMS".

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "insert new" è stata selezionata da "Manage Information About Database Management System".

Post-condition: Un Nuovo DMS è stato inserito.

Flow of events:

1. Un form, che consentirà l'inserimento di un nuovo DMS, è adesso visualizzato.
2. L'administrator inserisce il nome del nuovo DMS.
3. Seleziona il bottone "Insert".
4. DMS Inserito.

Secondary scenario:

3. Clicca sul bottone "Cancel".
4. Si ritorna alla pagina precedente.

Use case name: "Modify DMS".

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "Modify" è stata selezionata da "Manage Information About Database Management System".

Post-condition: Il DMS selezionato è stato modificato.

Flow of events:

1. Un form, che consentirà la modifica del DMS selezionato, è adesso visualizzato, nel campo di testo dedicato all'inserimento del nuovo nome del DMS è presente il nome del vecchio DMS.
2. L'administrator Modifica ciò che è contenuto nel campo di testo.

3. Clicca sul bottone “Modify”.

4. DMS Modificato.

Secondary scenario:

3. Clicca sul bottone “Cancel”.

4. Si ritorna alla pagina precedente.

Use case name: “Delete DMS”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: l'azione “Delete” è stata selezionata da “Manage Information About Database Management System”.

Post-condition: Il DMS selezionato è stato cancellato.

Flow of events:

1. Una finestra di dialogo, che chiede conferma per la futura cancellazione del DMS selezionato, è adesso visualizzata.

2. L'administrator clicca sul bottone “yes”.

3. Il DMS selezionato è stato cancellato.

Secondary scenario:

2. Clicca sul bottone “no”.

3. Si ritorna alla pagina precedente.

Use case name: “Manage Information About Technology”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: Nell'area di amministrazione è stata selezionato il link a “Manage Technology”.

Use case “included”: “Insert New Technology”, “Modify Technology”, “Delete Technology”.

Flow of events:

1. Una pagina nella quale sono visualizzate tutte le tecnologie disponibili, è adesso visualizzata.

2. Su ogni tecnologia esistente si possono effettuare due operazioni, a scelta: modifica e cancellazione.

3. Si può eventualmente inserire una nuova tecnologia.

4. Se viene scelta l'azione:

a. “insert new”, il controllo passa al caso d'uso “Insert New Technology”,

b. “modify”, in corrispondenza di un tecnologia già esistente, il controllo passa al

caso d'uso "Modify Technology", al quale viene passato anche la tecnologia da modificare;

- c. "delete", in corrispondenza di una tecnologia già esistente, il controllo passa al caso d'uso "Delete Technology", al quale viene passato anche la tecnologia da cancellare.

Use case name: "Insert New Technology".

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "insert new" è stata selezionata da "Manage Information About Technology".

Post-condition: Una nuova Technology è stata inserita.

Flow of events:

1. Un form, che consentirà l'inserimento di una nuova tecnologia, è adesso visualizzato.
2. L'administrator inserisce il nome della nuova tecnologia.
3. Clicca sul bottone "Insert".
4. Tecnologia inserita.

Secondary scenario:

3. Clicca sul bottone "Cancel".
4. Si ritorna alla pagina precedente.

Use case name: "Modify Technology".

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "Modify" è stata selezionata da "Manage Information About Technology".

Post-condition: La tecnologia selezionata è stata modificata.

Flow of events:

1. Un form, che consentirà la modifica della tecnologia selezionata, è adesso visualizzato, nel campo di testo dedicato all'inserimento del nuovo nome della tecnologia è presente il nome della vecchia tecnologia.
2. L'administrator Modifica ciò che è contenuto nel campo di testo.
3. Clicca sul bottone "Modify".
4. Tecnologia modificata.

Secondary scenario:

3. Clicca sul bottone "Cancel".

4. Si ritorna alla pagina precedente.

Use case name: “Delete Technology”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’azione “Delete” è stata selezionata da “Manage Information About Technology”.

Post-condition: La tecnologia selezionata è stata cancellata.

Flow of events:

1. Una finestra di dialogo, che chiede conferma per la futura cancellazione della tecnologia selezionata, è adesso visualizzata.
2. L’administrator clicca sul bottone “yes”.
3. La tecnologia selezionata è stata cancellata.

Secondary scenario:

2. L’administrator clicca sul bottone “no”.
3. Si ritorna alla pagina precedente.

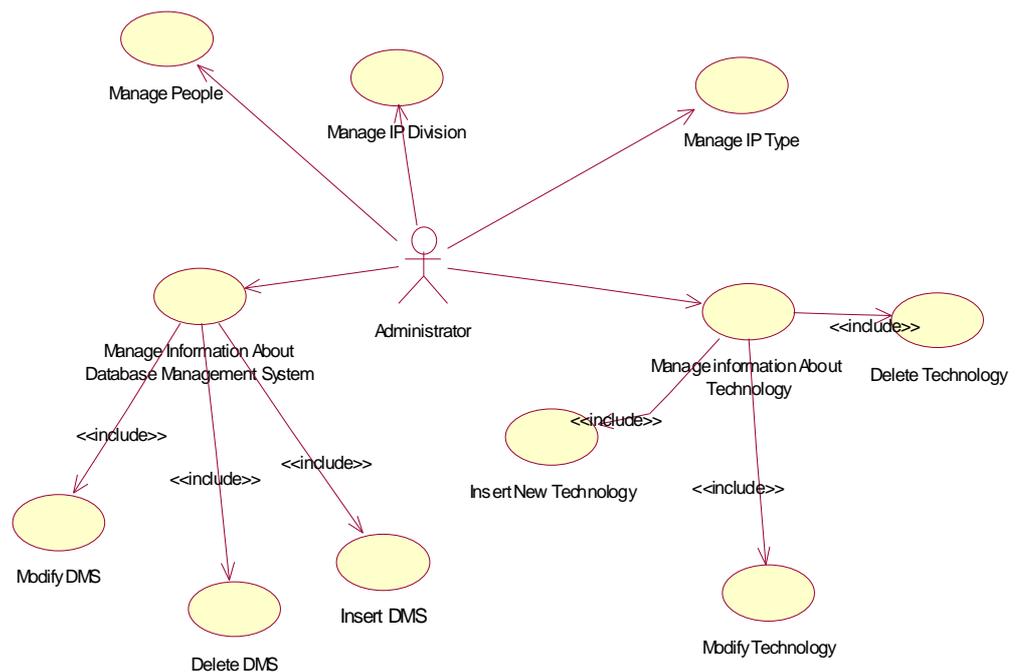


Figura 5.4 Diagramma dei casi d’uso (Admin IP Exchange)

In fine, analizziamo i diagrammi mostrati in figura 5.5. Essi rappresentano un raffinamento dei casi d'uso "Manage People", "Manage IP Division", "Manage IP Type" visti in figura 5.4.

Use case name: "Manage People".
Actors: Administrator.
Priority: 1.
Status: Requirements capture.
Pre-condition: Nell'area di amministrazione è stata selezionato il link "Manage People".
Post-condition:
Use case "included": "Insert New People", "Modify People", "Delete People".
Flow of events:

1. Viene visualizzata una pagina nella quale sono visualizzate tutte le persone appartenenti alle divisioni fornitrici (Provider/ Designer) disponibili.
2. Su ogni persona esistente si possono effettuare due operazioni : modifica e cancellazione.
3. Si può eventualmente inserire una nuova persona.
4. Se viene scelta l'azione:
 - a. "insert new people", il controllo passa al caso d'uso "Insert New People";
 - b. "modify", in corrispondenza di una persona già esistente, il controllo passa al caso d'uso "Modify People", al quale viene passato anche l'identificativo della persona da modificare.
 - c. "delete", in corrispondenza di una persona già esistente, il controllo passa al caso d'uso "Delete People", al quale viene passato anche l'identificativo della persona da modificare.

Use case name: "Insert NewPeople".
Actors: Administrator.
Priority: 1.
Status: Requirements capture.
Pre-condition: L'azione "insert new people" è stata selezionata da "Manage People".
Post-condition: Un nuovo provider/designer è stato inserito.
Use case "included": "find info ST people".
Flow of events:

1. Un form, che consentirà l'inserimento dei dati di una nuova persona, è adesso visualizzato.
2. Il form conterrà dei campi per l'inserzione dell'indirizzo e-mail della persona da caricare, della divisione di appartenenza, e due checkbox per indicare se la persona in questione è

un provider o un designer, oppure svolge entrambi i ruoli.

3. L'administrator inserisce i valori nei relativi campi.
4. Clicca sul bottone "Insert".
5. L'indirizzo e-mail inserito nel form viene passato al caso d'uso "find info ST people" il quale andrà a ricercare le informazioni relative nella Enterprise Directory.
6. "Find info ST people" ha trovato i dati relativi all'indirizzo e-mail (unico per ogni dipendente ST).
7. I dati relativi alla nuova persona (nome, cognome, divisione, gruppo, città, telefono, e-mail, designer, provider) possono adesso essere memorizzati.

Secondary scenario:

6. "Find info ST people" non ha trovato i dati relativi all'indirizzo e-mail .
7. Non è possibile proseguire nella registrazione di questa nuova persona.

Use case name: "Modify People".

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "Modify" è stata selezionata da "Manage People".

Post-condition: La persona selezionata è stata modificata.

Use case "included": "Find info ST people".

Flow of events:

1. Un form, che consentirà la modifica dei dati della persona selezionata, è adesso visualizzato, tutti dati già memorizzati per questa persona sono visibili, (nome, cognome, divisione, gruppo, città, telefono, e-mail, designer, provider).
2. L'administrator modifica i vari campi.
3. Clicca sul bottone "Modify".
4. Se sono stati modificati i campi "divisione" o "provider" (da vero diventa falso) si modifica il riferimento al tipo, se provider di un tipo particolare, o alla divisione se manager di una divisione.
5. Se viene modificato il campo e-mail il controllo passa a "find info ST people".

Secondary scenario:

3. Clicca sul bottone "Cancel".
4. Si ritorna alla pagina precedente.

Use case name: "Delete People".

Actors: Administrator.

Priority: 1.

<p>Status: Requirements capture.</p> <p>Pre-condition: L'azione "Delete" è stata selezionata da "Manage People".</p> <p>Post-condition: La persona selezionata è stata cancellata.</p> <p>Flow of events:</p> <ol style="list-style-type: none"> 1. Una finestra di dialogo, che chiede conferma per la futura cancellazione della persona selezionata, è adesso visualizzata. 2. L'administrator clicca sul bottone "yes". 3. La persona selezionata è stata cancellata. 4. Se la persona era un provider manager, viene cancellato il riferimento alla divisione, quindi il manager della divisione viene cancellato. 5. Se la persona era un provider, viene cancellato il riferimento al tipo da lui fornito. <p>Secondary scenario:</p> <ol style="list-style-type: none"> 2. Clicca sul bottone "no". 3. Si ritorna alla pagina precedente.

<p>Use case name: "Manage IP Division".</p> <p>Actors: Administrator.</p> <p>Priority: 1.</p> <p>Status: Requirements capture.</p> <p>Pre-condition: Nell'area di amministrazione è stata selezionato il link a "Manage IP Division".</p> <p>Use case "included": "Insert New IP Division", "Modify IP Division", "Delete IP Division".</p> <p>Flow of events:</p> <ol style="list-style-type: none"> 1. Viene visualizzata una pagina nella quale sono visualizzate tutte le divisioni ST che forniscono IP, con il relativo Provider Manager. 2. Su ogni divisione esistente si possono effettuare due operazioni: modifica e cancellazione. 3. Si può eventualmente inserire una nuova IP Division. 4. Se viene scelta l'azione: <ol style="list-style-type: none"> a. "insert new IP Division", il controllo passa al caso d'uso "Insert New IP Division"; b. "modify", in corrispondenza di una divisione già esistente, il controllo passa al caso d'uso "Modify IP Division", al quale viene passato anche la divisione da modificare; c. "delete", in corrispondenza di una divisione già esistente, il controllo passa al caso d'uso "Delete IP division", al quale viene passato anche la divisione da modificare.

Use case name: “Insert New IP Division”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’azione “insert new IP Division” è stata selezionata da “Manage IP Division”.

Post-condition: Una nuova IP Division è stata inserita, insieme con il suo Provider Manager.

Use case “included”: “Insert Provider Manager”.

Flow of events:

1. Un form, che consentirà l’inserimento dei dati di una nuova divisione, è adesso visualizzata.
2. Il form conterrà dei campi per l’inserzione del nome della divisione e dell’indirizzo e-mail del Provider Manager .
3. L’administrator inserisce i valori nei relativi campi.
4. Clicca sul bottone “Insert”.
5. L’indirizzo e-mail inserito nel form viene passato al caso d’uso “Insert Provider Manager” il quale andrà a ricercare le informazioni relative al provider manager.
6. I dati relativi alla nuova divisione ed al provider manager possono adesso essere memorizzati.

Secondary scenario:

5. “Insert Provider Manager” non ha trovato i dati relativi all’indirizzo e-mail.
6. Non è possibile proseguire nella registrazione di questa nuova Divisione.

Use case name: “Modify IP Division”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’azione “Modify” è stata selezionata da “Manage IP Division”.

Post-condition: L’ IP Division selezionata è stata modificata.

Use case “included”: “Insert provider Manager”.

Flow of events:

1. Un form, che consentirà la modifica dei dati della divisione selezionata, è adesso visualizzato, nel quale sono visibili la divisione da modificare, ed il nome del provider manager.
2. L’administrator modifica i vari campi.
3. Clicca sul bottone “Modify”.

4. Se il campo “provider manager” è stato modificato il manager attuale viene cancellato ed il controllo passa a “Insert provider Manager”.
 5. Se viene modificato il campo “IP division” si memorizza questa modifica.
 6. Vengono modificati anche : le People, gli IP_type, le Request IP che hanno un collegamento con la divisione modificata.
- Secondary scenario:**
3. Clicca sul bottone “Cancel”.
 4. Si ritorna alla pagina precedente.

- Use case name:** “Delete Division”.
- Actors:** Administrator.
- Priority:** 1.
- Status:** Requirements capture.
- Pre-condition:** L’azione “Delete” è stata selezionata da “Manage IP Division”.
- Post-condition:** La divisione selezionata è stata cancellata.
- Flow of events:**
1. Una finestra di dialogo, che chiede conferma per la futura cancellazione della divisione selezionata, è adesso visualizzata.
 2. L’administrator clicca sul bottone “yes”.
 3. La divisione selezionata è stata cancellata.
 4. Vengono anche cancellati: Provider, Provider Manager, People, IP_type relativi alla IP Division cancellata.
- Secondary scenario:**
2. Clicca sul bottone “no”.
 3. Si ritorna alla pagina precedente.

- Use case name:** “Manage IP Type”.
- Actors:** Administrator.
- Priority:** 1.
- Status:** Requirements capture.
- Pre-condition:** Nell’area di amministrazione è stata selezionato il link a “Manage IP Type”.
- Use case “included”:** “Insert New IP Type”, “Modify IP Type”, “Delete IP Type”.
- Flow of events:**
1. Viene visualizzata una pagina nella quale per ogni divisione, sono visualizzati tutti i tipi di IP forniti ed i relativi Provider.
 2. Su ogni tipo di IP esistente si possono effettuare due operazioni: modifica e cancellazione.
 3. Si può eventualmente inserire un nuovo tipo di IP.

4. Se viene scelta l'azione:
- “Insert new IP Type”, il controllo passa al caso d'uso “Insert New IP Type”.
 - “Modify”, in corrispondenza di un tipo di IP già esistente, il controllo passa al caso d'uso “Modify IP Type”, al quale viene passato anche il tipo di IP da modificare e la divisione relativa.
 - “Delete”, in corrispondenza di un tipo di IP già esistente, il controllo passa al caso d'uso “Delete IP Type”, anche il tipo di IP da cancellare e la divisione relativa

Use case name: “Insert New IP Type”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione “Insert new IP Type” è stata selezionata da “Manage IP Type”.

Post-condition: Una nuova IP Type è stata inserita, insieme con il suo Provider.

Use case “included”: “Insert Provider”.

Flow of events:

- Un form, che consentirà l'inserimento dei dati di un nuovo tipo di IP, per una divisione già esistente è adesso visualizzato.
- Il form conterrà dei campi per la scelta della divisione che intende fornire questo nuovo tipo, un campo per l'inserimento del tipo di IP ed uno per l'inserimento dell'indirizzo e-mail del Provider.
- L'administrator inserisce i valori nei relativi campi.
- Clicca sul bottone “Insert”.
- L'indirizzo e-mail inserito nel form viene passato al caso d'uso “Insert Provider” il quale andrà a ricercare le informazioni relative al Provider.
- I dati relativi al nuovo tipo di IP ed al provider possono adesso essere memorizzati.

Secondary scenario:

- “Insert Provider” non ha trovato i dati relativi all'indirizzo e-mail.
- Non è possibile proseguire nella registrazione del provider.
- L'IP Type verrà inserito lo stesso.

Use case name: “Modify IP Type”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione “Modify” è stata selezionata da “Manage IP Type”.

Post-condition: Il tipo selezionato è stato modificato.

Use case “included”: “Insert provider”.

Flow of events:

- Un form, che consentirà la modifica dei dati del tipo di IP selezionato, è adesso

visualizzato, nel quale sono visibili la divisione, il tipo di IP da modificare, ed il nome del Provider.

2. L'administrator modifica i vari campi.
3. Clicca sul bottone "Modify".
4. Se il campo "provider" è stato modificato il manager attuale viene cancellato ed il controllo passa a "Insert provider".
5. Se viene modificato il campo IP Type si memorizza questa modifica.
6. Vengono modificate anche le richieste che hanno un collegamento con il tipo di IP modificato.

Secondary scenario:

3. Clicca sul bottone "Cancel".
4. Si ritorna alla pagina precedente.

Use case name: "Delete IP Type".

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "Delete" è stata selezionata da "Manage IP Type".

Post-condition: L'IP Type selezionato è stato cancellato.

Flow of events:

1. Una finestra di dialogo, che chiede conferma per la futura cancellazione del tipo di IP selezionato, è adesso visualizzata.
2. L'administrator clicca sul bottone "yes".
3. L'IP Type selezionato è stato cancellato.
4. Il Provider relativo al tipo di IP, verrà cancellato come provider ma, rimarrà memorizzato come People.

Secondary scenario:

2. Clicca sul bottone "no".
3. Si ritorna alla pagina precedente.

Use case name: "Find ST people info".

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'indirizzo e-mail della persona da cercare è stato fornito dal caso d'uso chiamante.

Post-condition: Le informazioni sono state recuperate.

Nota: Le ricerche nella Enterprise Directory vengono fatte utilizzando come chiave di ricerca l'indirizzo e-mail perché esso è unico per ogni dipendente ST.

Flow of events:

1. Il caso d'uso è stato richiamato da un altro caso d'uso, il quale ha bisogno per la conclusione del suo lavoro delle informazioni di un dipendente ST avente come indirizzo e-mail quello fornito.
2. Le Informazioni dei dipendenti ST sono tutte contenute nella Enterprise Directory.
3. Il sistema si collega al server ldap;
4. Si ricerca il record che contiene l'indirizzo e-mail fornito dal caso d'uso chiamante.
5. Il record è stato individuato.
6. Si prelevano da questo record le seguenti Informazioni:
 - a) Nome.
 - b) Cognome.
 - c) Gruppo.
 - d) Divisione.
 - e) Città.
 - f) Telefono.
 - g) E-mail.
7. Queste informazioni vengono ritornate al caso d'uso chiamante.

Secondary scenario:

- A. 3. Il server ldap non è disponibile.
 4. Le informazioni non potranno essere recuperate.
- B. 3. L'indirizzo e-mail fornito è sbagliato.
 4. Non è stato individuato nessun record, le informazioni non potranno essere recuperate.

Use case name: "Insert Provider Manager".**Actors:** Administrator.**Priority:** 1.**Status:** Requirements capture.**Pre-condition:** L'indirizzo e-mail del provider manager da inserire è stato fornito dal caso d'uso chiamante.**Post-condition:** Un nuovo provider manager è stato inserito.**Use case "included":** "find info ST people".**Flow of events:**

1. Se fra le persone appartenenti alle divisioni esistenti vi è già l'indirizzo e-mail del candidato "provider manager".
2. Viene segnata come "provider manager" della divisione in questione.

Secondary scenario:

- A. 1. Se fra le persone appartenenti alle divisioni esistenti non vi è l'indirizzo e-mail del candidato "provider manager". Il controllo passa a "find info ST people" il quale andrà a ricercare le informazioni relative nell'Enterprise Directory.
 2. "find info ST people" ha trovato i dati relativi all'indirizzo e-mail (unico per ogni dipendente ST).
 3. I dati relativi alla nuova persona (nome, cognome, divisione, gruppo, città, telefono, e-

mail, designer, provider) possono adesso essere memorizzati.

4. La people appena memorizzata viene segnata come “provider manger”
divisione in questione.

- B. 2. “find info ST people” non ha trovato i dati relativi all’indirizzo e-mail .
3. Non è possibile proseguire nella registrazione di questa nuova people.
4. Non si può memorizzare il “provider manager”.

Use case name: “Insert Provider”.

Actors: Administrator.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’indirizzo e-mail del provider da inserire è stato fornito dal caso d’uso chiamante.

Post-condition: Una nuovo provider è stato inserito.

Use case “included”: “find info ST people”.

Flow of events:

1. Se fra le persone appartenenti alle divisioni fornitrici esistenti, vi è già l’indirizzo e-mail del candidato “provider”.
2. La persona in questione viene segnata come “provider” per il tipo di IP della divisione in questione.

Secondary scenario:

- A. 1. Se fra le “People” esistenti non vi è l’indirizzo e-mail del candidato “provider”.
2. Il controllo viene passato a “find info ST people” il quale andrà a ricercare le informazioni relative nell’ Enterprise Directory.
3. “find info ST people” ha trovato i dati relativi all’indirizzo e-mail (unico per ogni dipendente ST).
4. I dati relativi alla nuova persona (nome, cognome, divisione, gruppo, città, telefono, e-mail, designer, provider) possono adesso essere memorizzati.
5. La persona appena memorizzata viene segnata come “provider” per il tipo e divisione selezionati.
- B. 3. “find info ST people” non ha trovato i dati relativi all’indirizzo e-mail .
4. Non è possibile proseguire nella registrazione di questa nuova people.
5. Non si può memorizzare il “provider ”

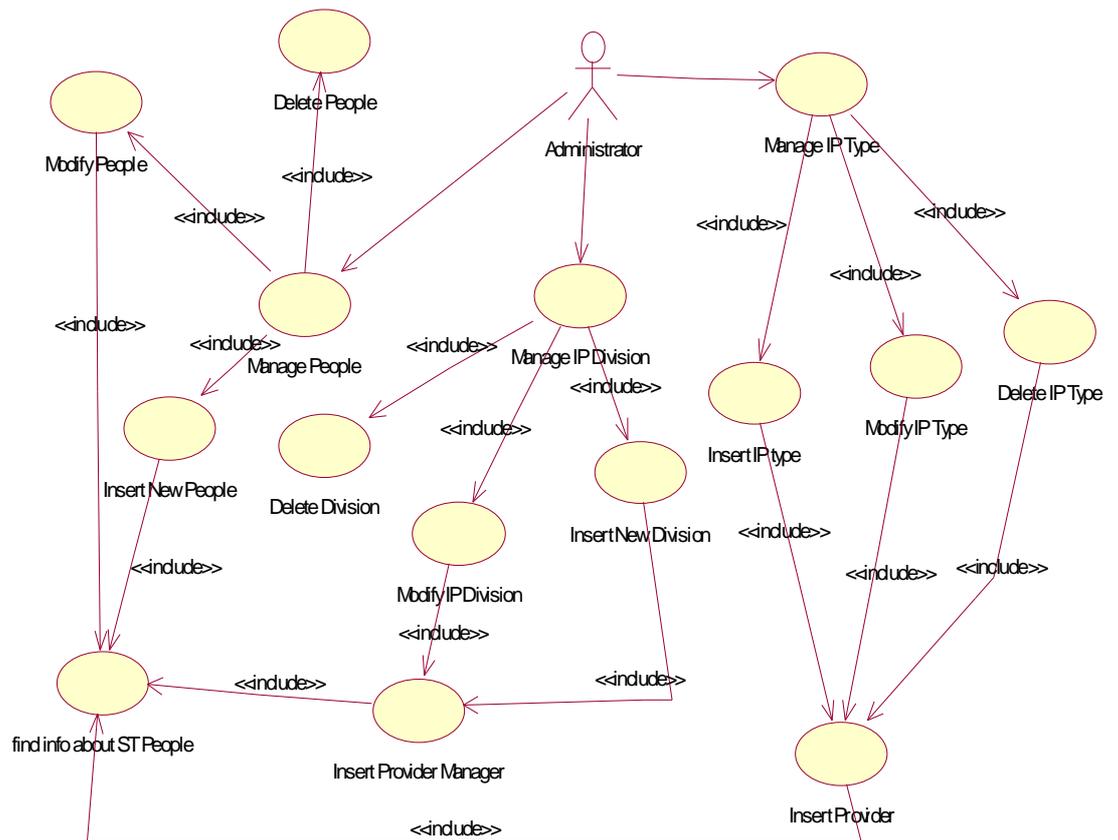


Figura 5.5 Diagramma dei casi d'uso (AdminIP Exchange)

6 Analisi e Design dell'applicazione “IP Exchange”

In questo paragrafo verranno analizzati i diagrammi che mostrano le caratteristiche della nostra applicazione sotto vari punti di vista. Saranno esposti i diagrammi: concettuale, di navigazione, e di presentazione. Per la costruzione di questi diagrammi è stata adottata la metodologia introdotta nel Capitolo 1.

6.1 Diagramma delle classi “type”

Per lo sviluppo di IP Exchange sono stati definiti dei nuovi tipi, essi sono rappresentati in figura 5.6.

Il tipo Info_people definisce tutti i dati necessari per identificare un dipendente ST, queste informazioni sono necessarie per il funzionamento di IP Exchange. Gli elementi appartenenti a questo tipo dipendono dall'Enterprise Directory (descritta nel capitolo 4).

Il Tipo IP è stato introdotto per modellare tutte le informazioni necessarie per identificare un IP. E' stato creato il tipo IP strutturato come in figura 5.6 in previsione di future implementazioni delle procedure ECR, NCN, ecc.

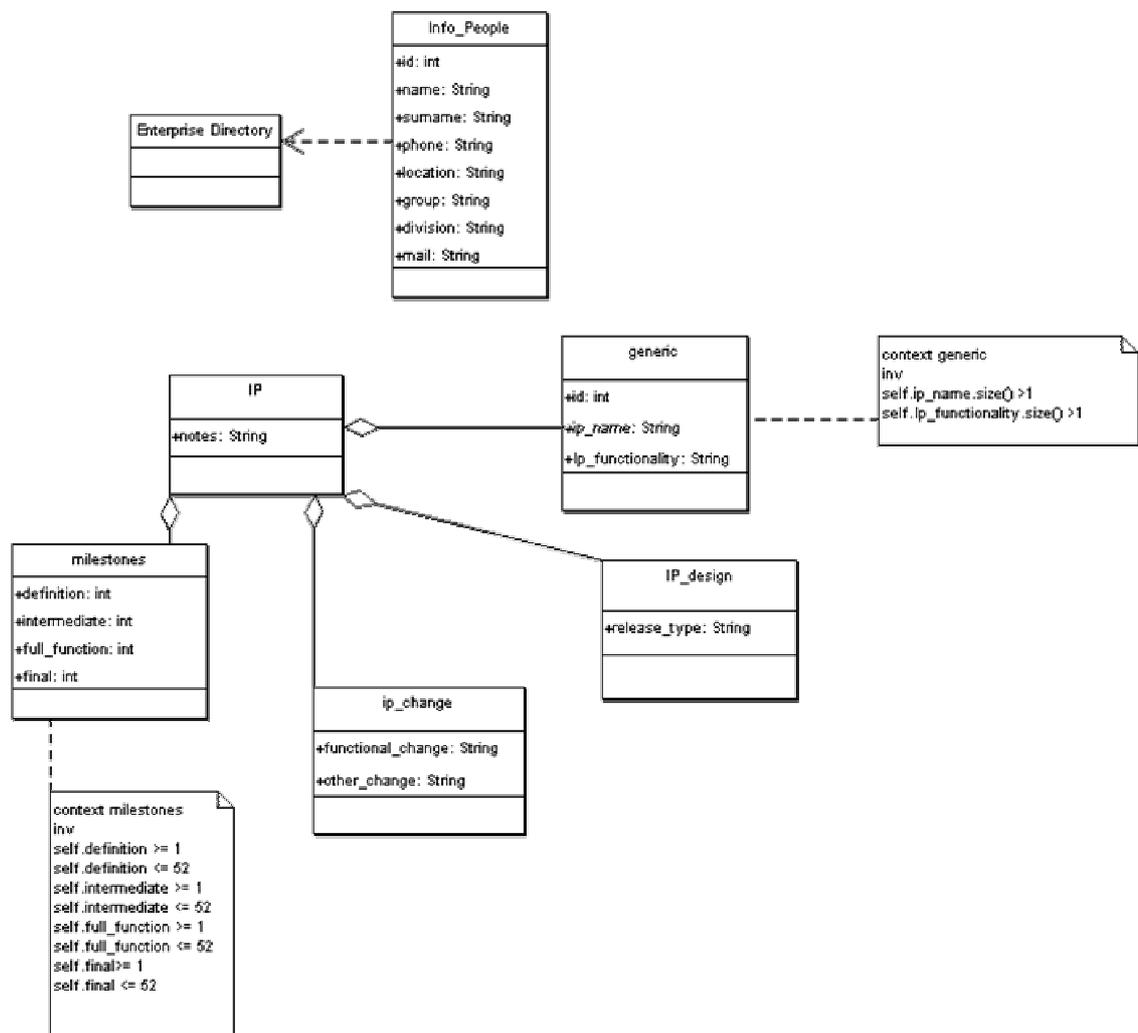


Figura 5.6 Diagramma dei nuovi tipi definiti

6.2 Diagramma Concettuale

Il diagramma concettuale per IP Exchange è mostrato in figura 5.7. E' un diagramma del dominio del caso di studio. Include, quindi tutti i concetti che sono rilevanti per l'applicazione IP Exchange.

Saranno presentate tre viste diverse del diagramma concettuale di IP Exchange: *General view*, *User view*, *Session view*; visibili rispettivamente nelle figure 5.7, 5.8, 5.9.

Lo scopo principale dell'applicazione IP Exchange è l'automazione del flusso di comunicazione associato allo scambio degli IP fra i clienti ed i fornitori. Gli elementi essenziali saranno quindi:

1. I fornitori, modellati dalla classe "Division Provider".
 2. I clienti, modellati tramite la classe "Customer" (specializzazione della classe User, figura 5.8).
 3. Le richieste, modellate dalla classe "REQUEST IP".
 4. I database management system, modellati tramite la classe "DMSystem", rappresentano il modo in cui verranno spedite le varie versioni degli IP prodotti (versioni parziali o definitive).
 5. Le tecnologie disponibili per l'implementazione modellate tramite la classe "Technology".
- La tecnologia è un'informazione molto importante perché un IP "technology dependent" è progettato sulla tecnologia, cioè al variare della tecnologia varia il modo d'implementare l'IP. Le classi appena citate avranno una relazione di composizione con la classe IP Exchange.

Come già detto, la classe "Division Provider", modella le divisioni che forniscono IP, essa sarà identificata tramite un nome. Ogni "Division Provider" fornisce uno o più tipi diversi di IP. Come tipo di IP s'intende, ad esempio, audio, video, ecc. I tipi sono modellati dalla classe "Type of IP" e sono individuati tramite un nome. Fra "Division Provider" e "Type of IP" vi è un'associazione uno a molti, poiché una divisione può fornire più tipi, la coppia (divisione, tipo) è unica.

Le divisioni sono formate da persone. Le informazioni necessarie per la nostra applicazione, relative alle persone sono modellate tramite la classe "People" (presente in figura 5.8). Essa conterrà tutte informazioni di tipo "Info_People" definito dal diagramma delle classi "types".

Il Designer sarà scelto per lo sviluppo degli IP richiesti.

Ogni “Division Provider” pubblica un catalogo degli IP esistenti per ogni “Type Of IP”. Il catalogo è modellato dalla classe “CATALOGUES”⁹.

Un altro elemento fondamentale per lo scambio di IP è la richiesta. Essa è modellata dalla classe REQUEST IP.

I suoi attributi sono:

1. **Requested:** di tipo IP, il quale è stato definito nel diagramma delle classi “types”;
2. **product:** indica il chip dove l’IP sarà integrato;
3. **tape_ship:** è una data, indica la data presunta di rilascio del chip integrante;
4. **day_request:** tiene traccia del giorno in cui è stata inviata la richiesta;
5. **status:** tiene traccia dello stato della richiesta.

La richiesta deve essere inviata ad una divisione ben determinata per un tipo di IP specifico, inoltre, nella richiesta si può indicare anche il “DMSsystem” ed il “Technology” preferito. Fra “REQUEST IP” e le classi appena citate c’è un’associazione unidirezionale con cardinalità massima pari ad uno.

Dalla combinazione di “Division Provider” e “Type of IP” entrambi indicati nella richiesta si deriva il provider a cui verrà inviata la Notify, tramite questo vincolo OCL.

```
Context Request IP
Inv
notify.from=request.Customer.Info_people.mail;
notify.to=request.typeofip.provider.Info_people.mail;
notify.cc=request.division.providermanager.Info_people.mail;
```

Notify è una classe che modella la notifica che verrà mandata in seguito a delle determinate azioni.

⁹ Attualmente il catalogo è un documento pdf. Esso quindi potrà essere semplicemente visualizzato. E’ un documento a sè per questo motivo non sono state aggiunte ulteriori associazioni fra Division Provider e CATALOGUES, o CATALOGUES e Type Of IP. Gli attributi di questa classe servono solo ad distinguere un file dall’altro.

Per ogni richiesta, vi sarà al più una proposta, essa è modellata dalla classe “PROPOSAL”. Essa verrà compilata dal provider se la richiesta è stata accolta. I suoi attributi sono:

1. *Proposed*: di tipo IP;
2. *cost*: intero indicante il costo, in uomo/mese, dello sviluppo della richiesta;
3. *risk factor*: una stringa nella quale verranno indicati se esistono dei rischi per lo sviluppo;
4. *criteria*: di tipo stringa, indica gli eventuali controlli da eseguire preventivamente al rilascio;
5. *day_proposal* e *day_acceptance*: sono due date che indicano rispettivamente il giorno in cui è stata inviata la proposta ed il giorno in cui il customer la ha restituita (accettata o no);
6. *Start_work* ed *end_work*: indicano le date di inizio e fine lavori;
7. *status*: indica lo stato della proposta.

Dalla richiesta si derivano gli indirizzi e-mail del cliente e del fornitore, per inviare la Notify della proposta inviata, tramite questo vincolo OCL:

```
Context Proposal
Inv
notify.from=request.typeofip.provider.Info_people.mail;
notify.to =request.Customer.Info_people.mail;
notify.cc=request.division.providermanager.Info_people.mail;
```

Alla proposta possono essere associati dei *designer* selezionati per lo sviluppo. Se la proposta è stata accettata si passa alla fase di sviluppo o spedizione, in cui vengono consegnate al cliente le versioni (parziali o definitive) degli IP prodotti, queste sono modellate dalla classe “RELEASE”. Essa è associata alla classe “Proposal”. “RELEASE” non è comunque una classe rilevante per la navigazione perché è un oggetto che verrà inviato attraverso i *database management system*, cioè dei sistemi interni ST che prescindono dall'applicazione Web.

Se la richiesta ha raggiunto lo stato “completed”, su richiesta del provider, il customer dovrà inviare il suo *feedback* (questionario), per indicare se le sue aspettative sono state soddisfatte o meno. Il questionario è modellato dalla classe “Questionnaire”, questa è associata alla richiesta (“Request IP”) con molteplicità massima uguale ad uno.

La *User view* del diagramma concettuale, mira ad individuare le caratteristiche dei diversi utenti, esso è mostrato in figura 5.8. Nel nostro caso gli utenti sono tutti dei dipendenti ST, però hanno dei ruoli differenti e quindi possono effettuare delle operazioni differenti. Il

diagramma in figura 5.8, descrive le varie classi tramite le quali verranno modellate le informazioni necessarie per i vari tipi d'utente.

Un utente autenticato è modellato dalla classe User, la quale individua univocamente ogni utente tramite Username e Password. Questi attributi corrispondono ai campi del record dell'Enterprise Directory di ogni dipendente ST.

Uno User ha tre specializzazioni principali: Customer, Provider ed Administrator, i quali potranno effettuare delle operazioni differenti.

In particolare il "Customer" potrà inviare la richiesta, accettare o rifiutare la proposta, od inviare il suo feedback. Queste ultime azioni non saranno visibili nei diagrammi, poiché queste azioni verranno effettuate come risposta a determinati e-mail mandate tramite i vari metodi delle classi in questione, e non come risultato di una navigazione attraverso le pagine.

Il "Provider" ha a sua volta una specializzazione: "Manager". "Provider" e "Manager" potranno effettuare le stesse operazioni con l'unica differenza che il "Manager" accederà a tutti i dati riguardanti l'intera divisione da lui gestita. Mentre il "Provider" accederà esclusivamente ai dati riguardanti la divisione a cui appartiene ma ristretti ai tipi di IP da lui gestiti.

L'attore provider in particolare potrà analizzare le richieste, cambiare il loro stato, inviare le proposte, la notifica di release, ed il questionario finale al customer.

L'"administrator" invece gestirà tutti i dati necessari per il funzionamento dell'applicazione. Infatti, gestirà i dati di: "People", "Division Provider", "Type of IP", "DMSystem", "Technology".

La *Session view* mostra le associazioni fra "IP Exchange", la sessione corrente e l'utente (*current user*) di una determinata sessione. Questa vista modella delle informazioni che si avranno solo a tempo di esecuzione, ma che sono rilevanti per la modellazione sia concettuale sia di navigazione.

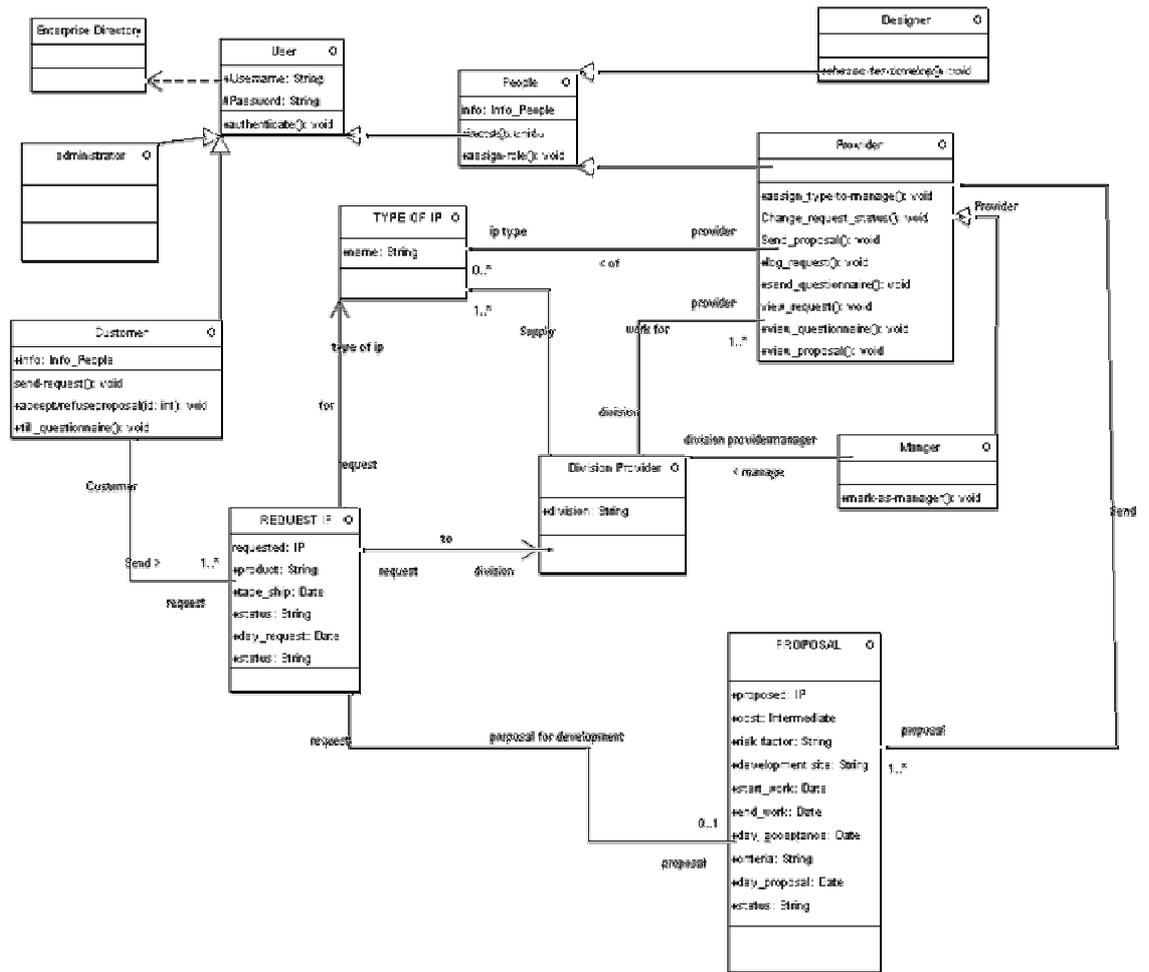


Figura 5.8 Diagramma Concettuale di IP Exchange (*User view*)

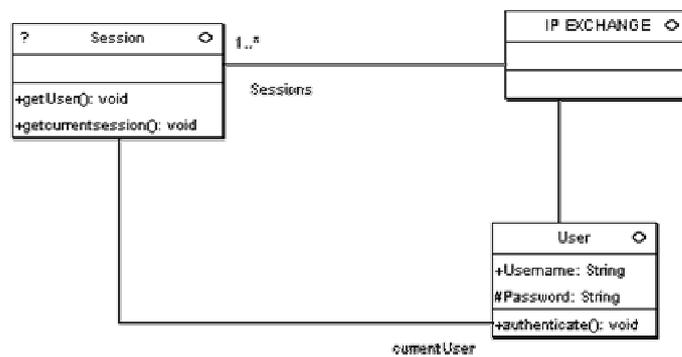


Figura 5.9 Diagramma Concettuale di IP Exchange (*Session view*)

6.3 Diagramma di Navigazione

La progettazione della navigazione definisce la struttura delle applicazioni Web ed indica come la navigazione potrà essere effettuata dall'utente.

Il modello di navigazione è costruito da due passi. Nel primo si definiscono *quali* oggetti possono essere potenzialmente raggiunti durante la navigazione, diagramma dello spazio di navigazione. Nel secondo si definisce *come* questi oggetti saranno raggiunti, diagramma della struttura di navigazione.

6.3.1 Diagramma dello Spazio di Navigazione

Il diagramma dello spazio di navigazione per IP Exchange è stato costruito seguendo il metodo indicato nel capitolo 1, saranno presentate tre viste poiché ogni attore accederà ad un'area di navigazione differente.

Le classi rilevanti per la navigazione sono: IP EXCHANGE, CATALOGUES, REQUEST IP, PROPOSAL, QUESTIONNAIRE, Division Provider, TYPE OF IP, People, DMSsystem, Technology.

Le classi che sono state omesse, perché non rilevanti sono: Notify e Release. Manager, Provider e Designer sono state omesse ma in People è stato aggiunto un attributo derivato *role* che indica il ruolo della "People".

Analizzando in dettaglio, la figura 5.10 raffigura il diagramma dello spazio di navigazione per un customer. Dall'*home page* (IP Exchange), un customer potrà visionare i cataloghi, le divisioni che forniscono gli IP, per ogni divisione potrà vedere chi sono i provider ed i tipi forniti, ed in oltre potrà inviare una nuova richiesta alla quale dovrà associare " Provider", "Type Of IP", "DMSsystem", "Technology".

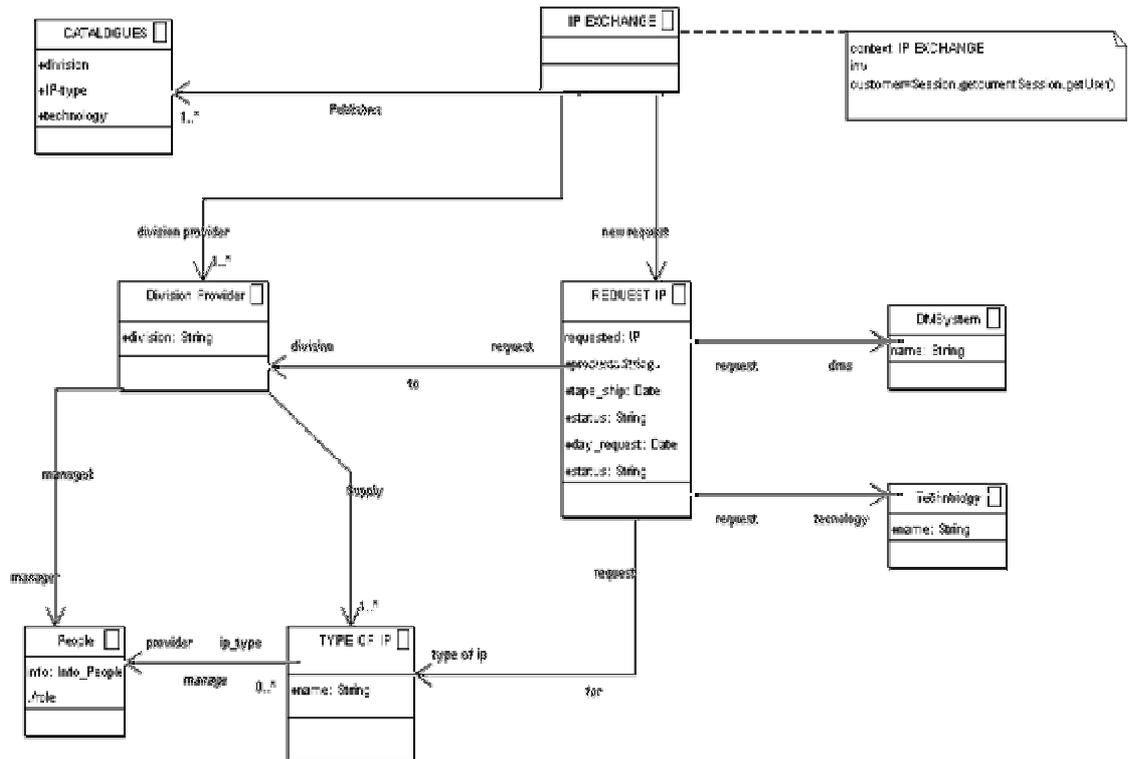


Figura 5.10 Diagramma dello Spazio di Navigazione (vista relativa al customer)

In figura 5.11 è presentato il diagramma dello spazio di navigazione per un provider o un manager. Tramite il quale potrà accedere alle richieste che soddisfano il seguente vincolo OCL.

```

Context IP Exchange
Inv "request presented depending on provider"
Let ur=Session.getCurrentSession().getUser()
If ur.ocIsTypeOf(Provider)
Then
request->excludesAll(r:REQUEST IP |
r.division.division<>ur.division.division
and r.typeofip.name<>ur.iptype.name)
else If ur.ocIsTypeOf(Manager)
request->
excludesAll(r:REQUESTIP |
r.division.division<>ur.manage.divisionprovider.division)
end if
end if

```


In figura 5.12 si può vedere la vista relativa ad un administrator. Dalla home page un amministratore autenticato, potrà gestire (inserire o modificare o cancellare) i dati relativi le seguenti classi: “DMSystem”, “Technology”, “Division Provider”, “Type Of IP”, e le “People”. Con l’associazione “update” s’indica la possibilità di cancellare o modificare un’istanza di una determinata classe.

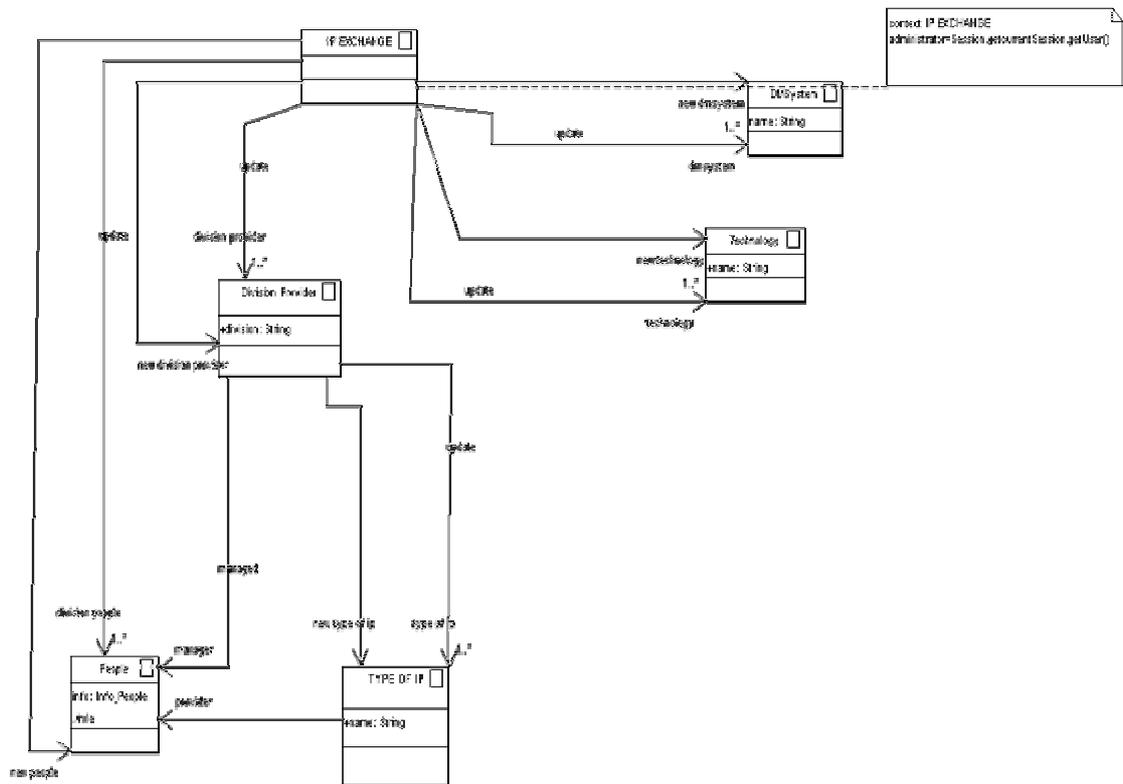


Figura 5.12 Diagramma dello Spazio di Navigazione (vista relativa all'administrator)

6.3.2 Diagramma della Struttura di Navigazione

I diagrammi dello spazio di navigazione sono stati migliorati, introducendo le primitive d’accesso, per ogni associazione diretta avente molteplicità maggiore di uno ed i menu per ogni classe che ha più associazioni uscenti.

In figura 5.13 si può vedere il diagramma della struttura di navigazione del provider.

Da “Manage IP Main Menu”, si possono effettuare tutte le azioni principali riguardanti la gestione degli IP.

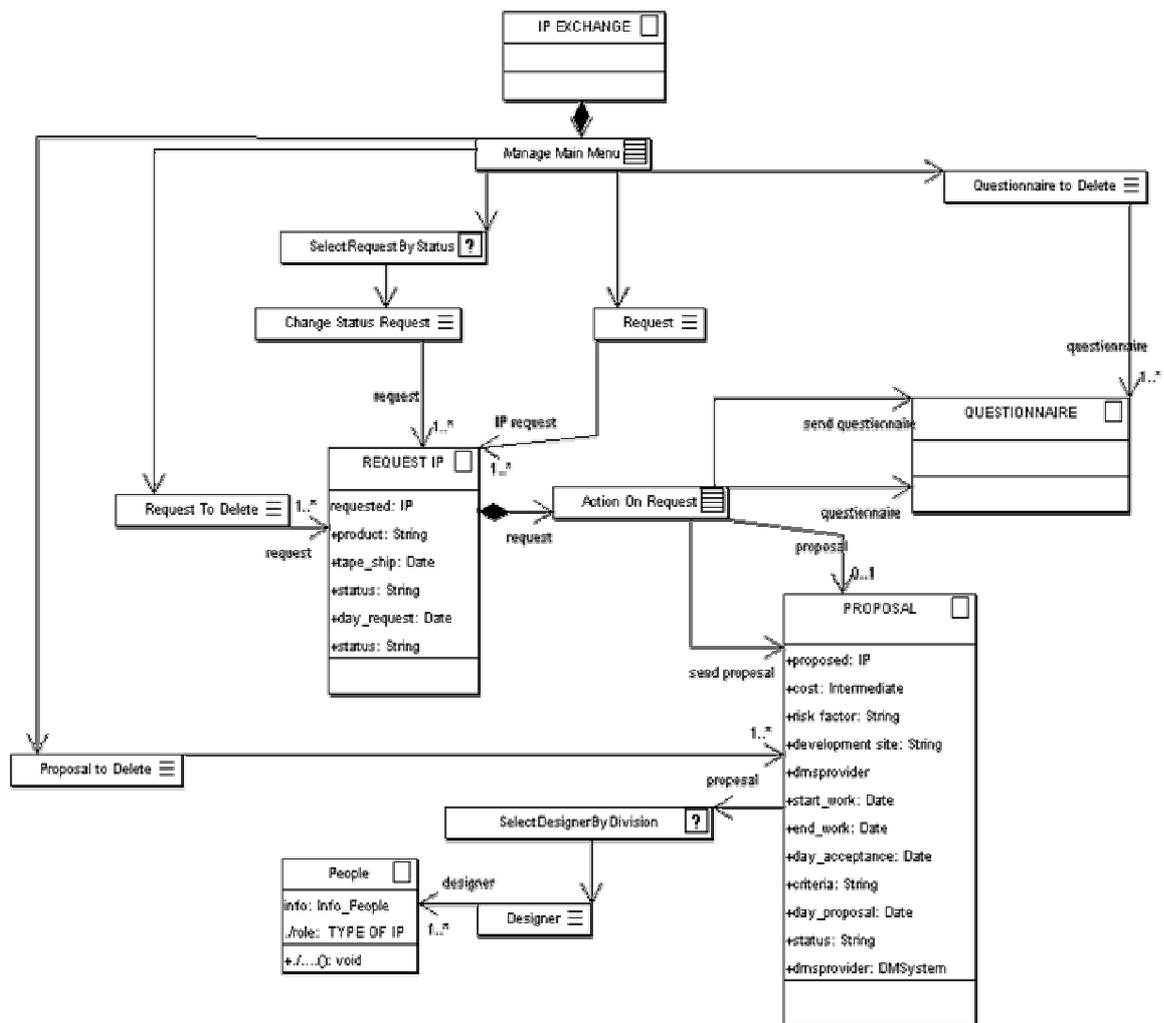


Figura 5.13 Diagramma della struttura di Navigazione (vista relativa al provider)

In figura 5.14, si pu  vedere il diagramma della struttura di navigazione della vista relativa al customer.

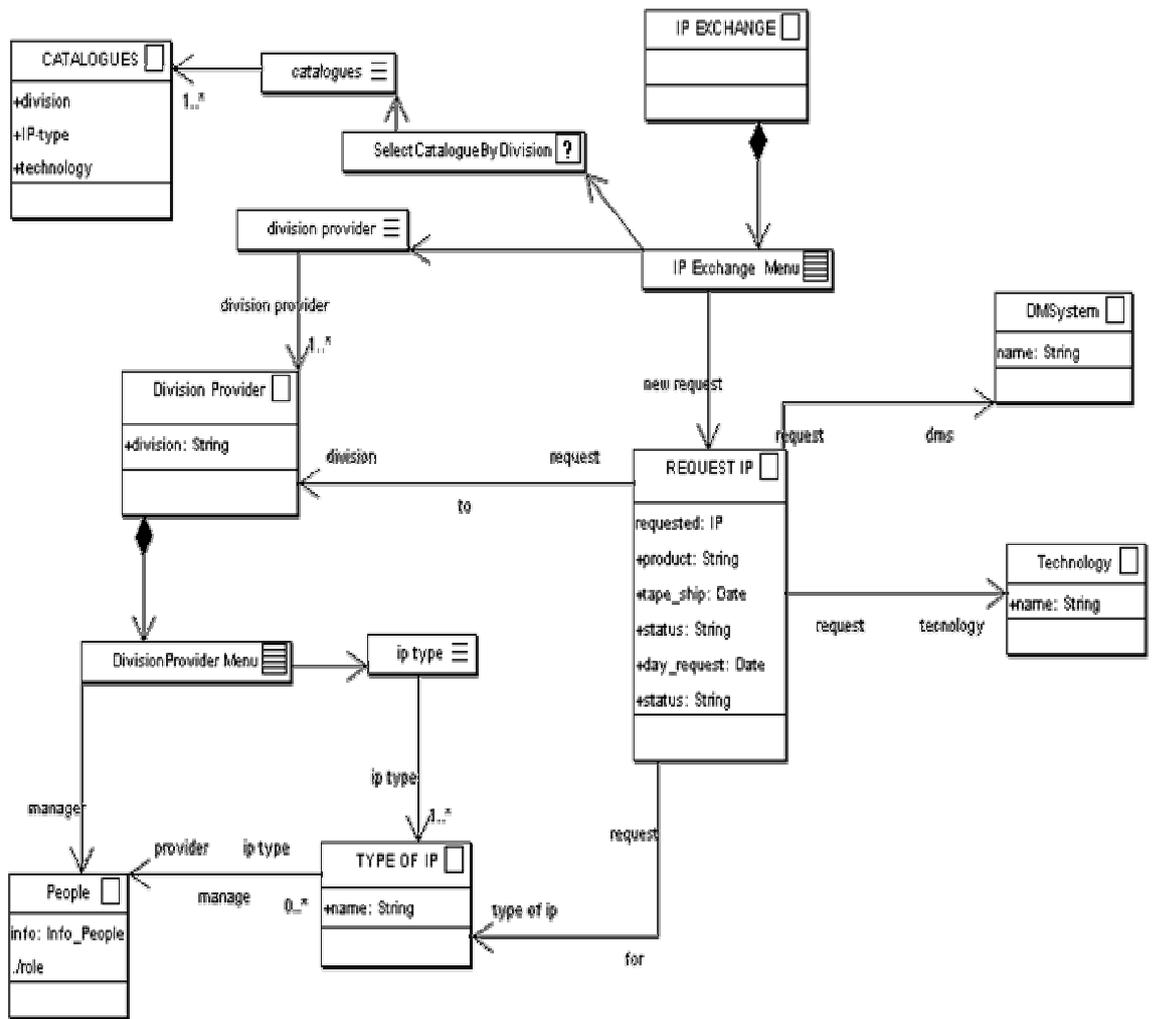


Figura 5.14 Diagramma della struttura di Navigazione (vista relativa al customer)

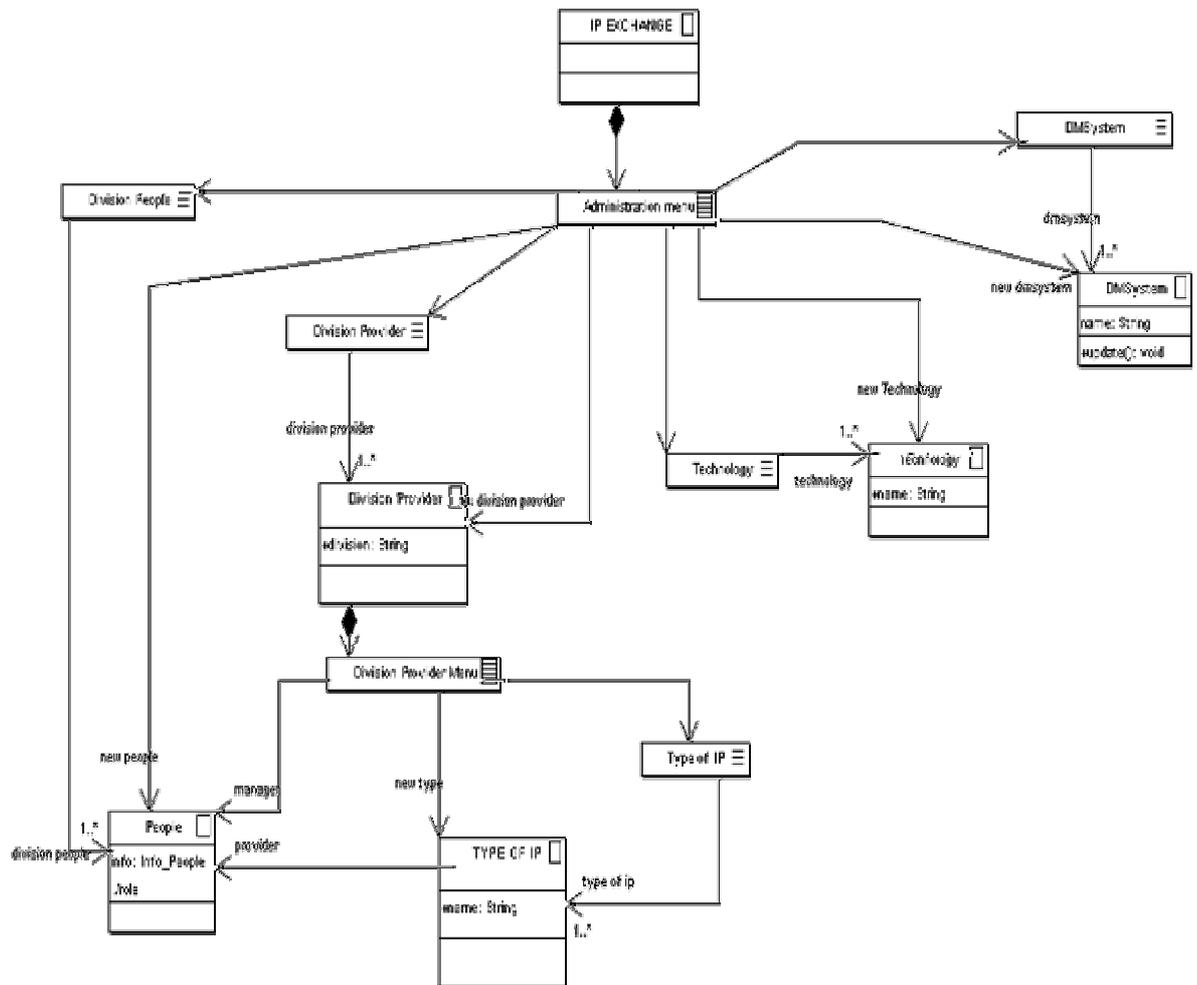


Figura 5.15 Diagramma della struttura di Navigazione (vista relativa all'administrator)

6.4 Diagramma di Presentazione

Dopo aver aggiunto i menu ed aver organizzato le associazioni nel modello di navigazione, si crea il diagramma di presentazione.

Secondo la metodologia mostrata nel capitolo 1, il modello di presentazione modella gli aspetti statici e dinamici delle presentazioni di un'applicazione Web.

Purtroppo, però, così come è stato detto nel capitolo 3, il tool attualmente disponibile per la metodologia UWE, ArgoUWE, non supporta la modellazione degli aspetti dinamici della presentazione.

In questo paragrafo quindi verranno mostrati solo gli aspetti presentazione a livello statico.

Si può dire, però, che concordemente alle richieste della divisione CMGDesign, si è scelto di utilizzare la tecnica dei frameset a finestra singola. Il frameset è suddiviso in due frame (sinistro e destro). Nel frame sinistro saranno presenti i menu, mentre in quello destro il contenuto delle classi di presentazione.

I dettagli riguardanti gli aspetti dinamici della presentazione sono stati concordati direttamente con la divisione CMGDesign, per essi non è stata prodotta però una documentazione scritta.

Nei diagrammi sottostanti, si possono vedere le classi di presentazione create per ogni classe di navigazione e per ogni attributo delle classi di navigazione. Le classi di presentazione per gli attributi sono connesse alle classi di presentazione della classe di navigazione a cui appartengono, tramite composizione. Per ogni menu viene creata una classe di presentazione, legata tramite composizione con la classe di navigazione che contiene il menu. Ad ogni "index" e "query" viene associata una classe di presentazione. In particolare, se un index od un query è puntato direttamente da un menù esso avrà una relazione di composizione con il menu per indicare che la pagina relativa sarà composta da un menu nella parte sinistra, e le relative presentazioni di index o query nella parte destra tramite le quali si accede alle classi di navigazioni, il cui contenuto è presentato dalla relativa classe di presentazione.

Lo scopo delle classi di presentazioni, secondo la metodologia mostrata nel capitolo 1, è mostrare degli scenari di presentazione. Come già detto esistono diversi tipi di presentazione degli scenari: *storyboarding*, *map-based*, *menu-based*. Il tool attualmente esistente non permette di effettuare le suddette presentazioni. Quindi presenteremo, semplicemente, le classi singole divise per viste così come è stato presentato il diagramma di navigazione.

Classi di presentazione per la vista relativa al provider.

Le classi di presentazione sottostanti sono relative al diagramma della struttura di navigazione mostrato in figura 5.13.

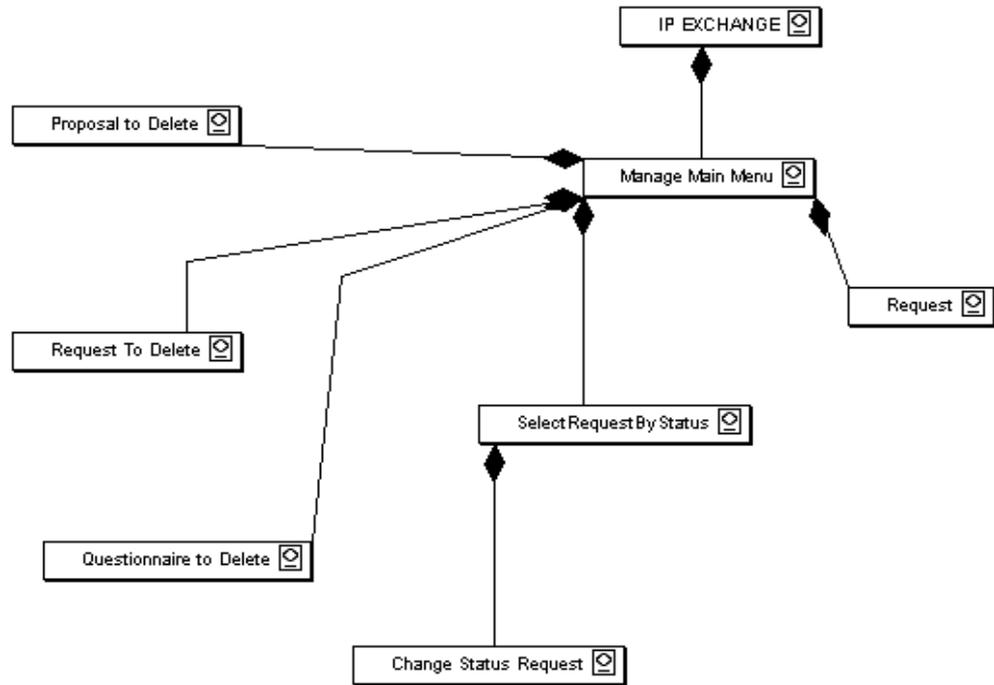


Figura 5.16. Diagramma di presentazione vista relativa al provider classe IP Exchange

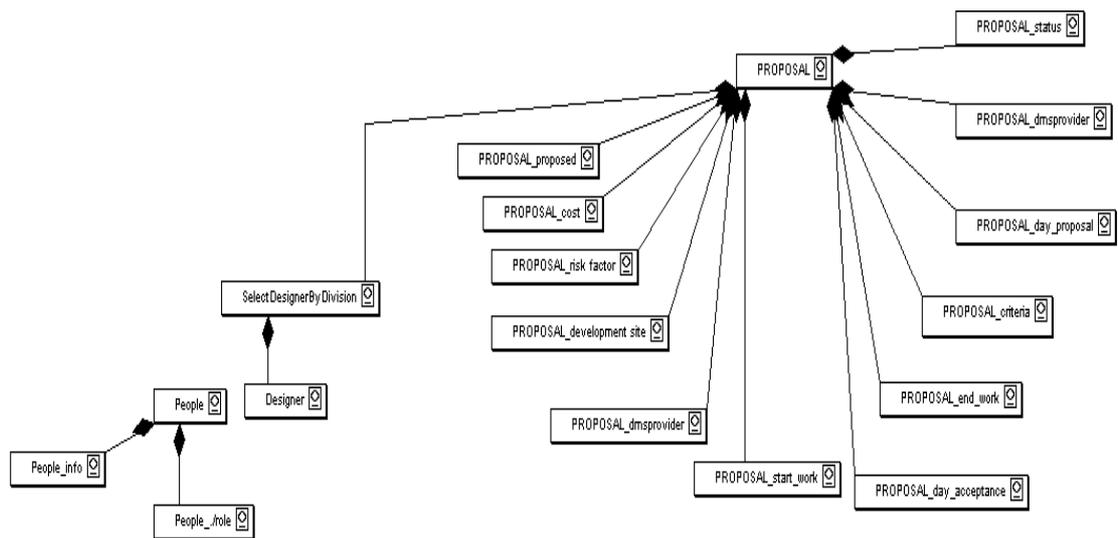


Figura 5.17. Diagramma di presentazione vista relativa al provider classe Proposal

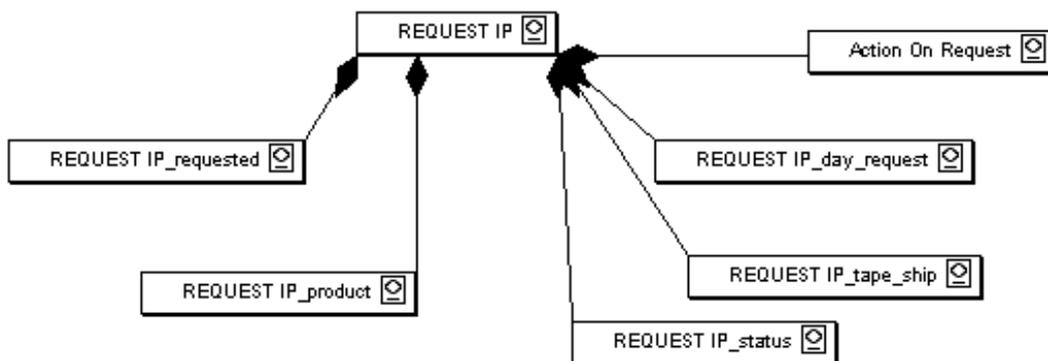


Figura 5.18. Diagramma di presentazione vista relativa al provider classe Request IP.

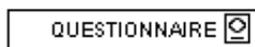


Figura 5.19. Diagramma di presentazione vista relativa al provider classe Questionnaire.

Classi di presentazione per la vista relativa al customer:

Le classi di presentazione sottostanti sono relative al diagramma della struttura di navigazione mostrato in figura 5.14:

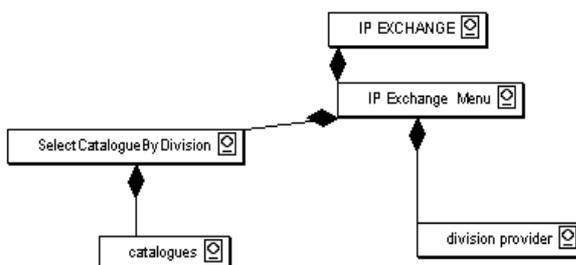


Figura 5.20. Diagramma di presentazione vista relativa al customer classe IP Exchange

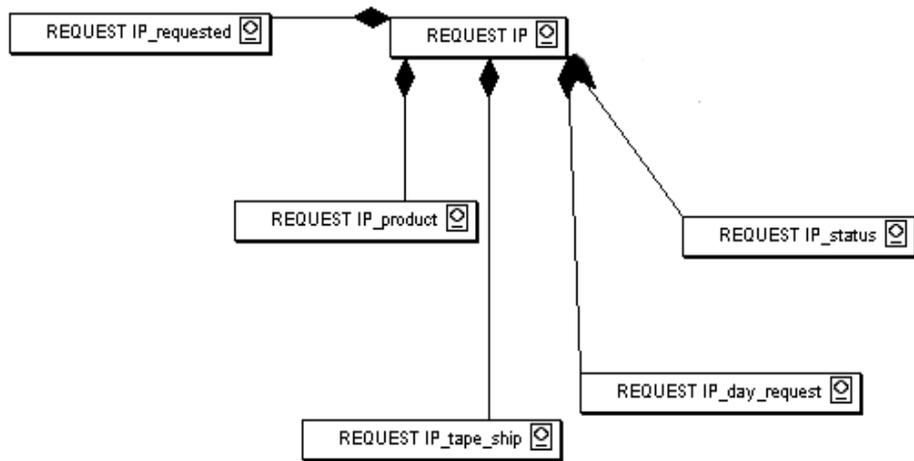


Figura 5.21. Diagramma di presentazione vista relativa al customer classe Request IP

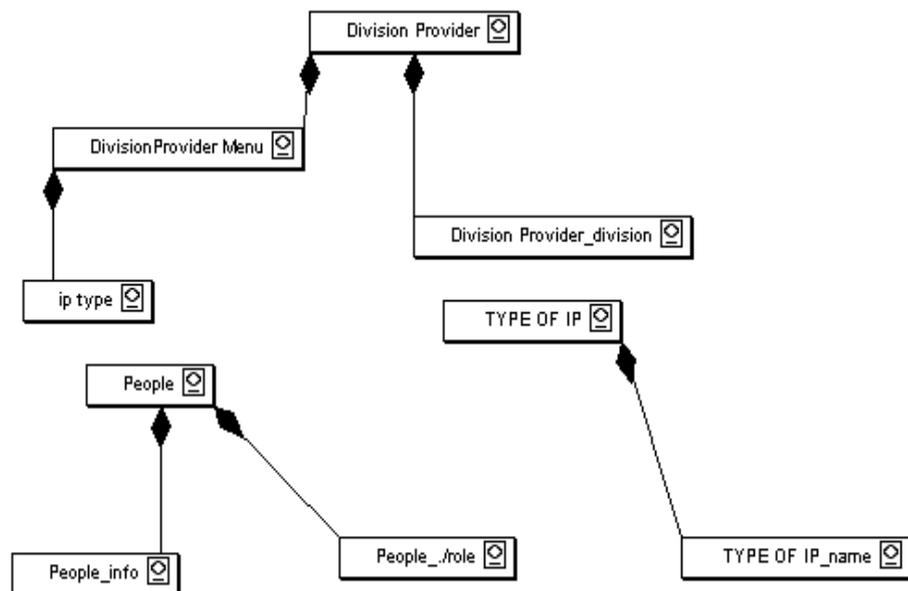


Figura 5.22. Diagramma di presentazione vista relativa al customer classi Division Provider, People e TYPE OF IP

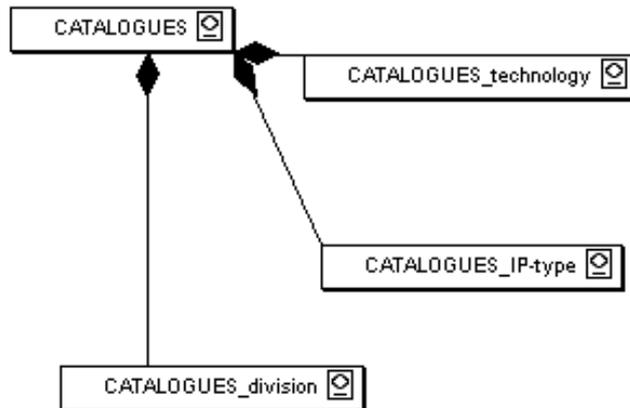


Figura 5.23. Diagramma di presentazione vista relativa al customer classe Catalogues

Classi di presentazione per la vista relativa al administrator:

Le classi di presentazione sottostanti sono relative al diagramma della struttura di navigazione mostrato in figura 5.15:

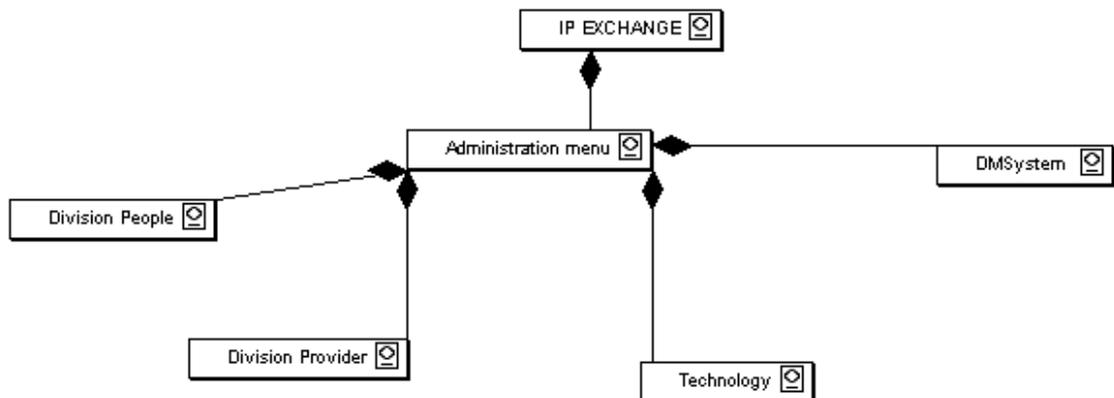


Figura 5.24. Diagramma di presentazione vista relativa all'administrator classe principale IP Exchange

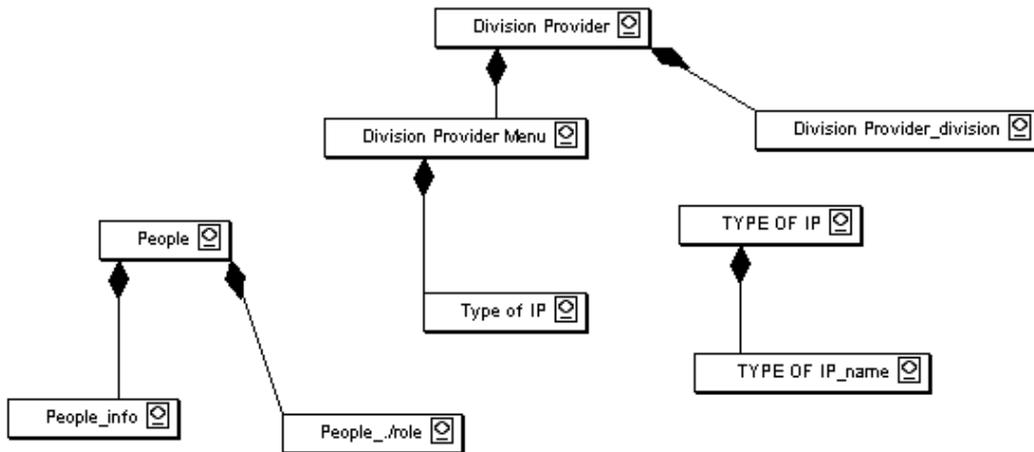


Figura 5.25. Diagramma di presentazione vista relativa all'administrator classe principale Division Provider, People, TypeOfIP.

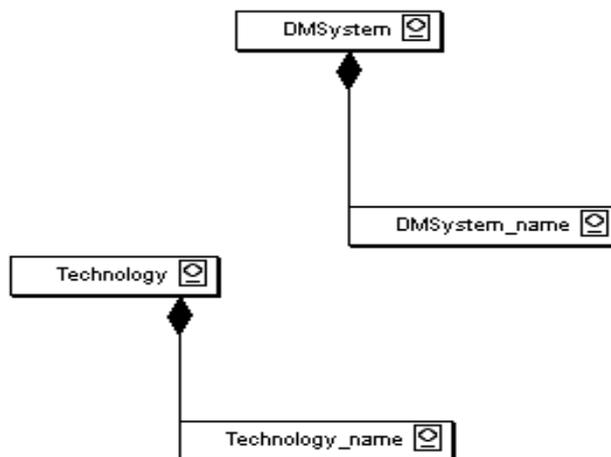


Figura 5.26. Diagramma di presentazione vista relativa all'administrator classi DMSystem e Technology

7 Progettazione della Struttura dell'applicazione

Lo scopo di questo tipo di progettazione è analizzare e modellare la struttura dell'applicazione nota a livello di progettazione, quindi prima dell'implementazione, definendo:

- Sottosistemi e loro interfacce.
- Progettazione delle classi, che sono rilevanti per la struttura.
- Meccanismi di progettazione generici per trattare requisiti funzionali e non funzionali.
- Possibilità di riutilizzo, di parti del sistema o di prodotti software in generale.

Uno schema di come sarà strutturata l'architettura di IP Exchange è mostrata in figura 5.27.

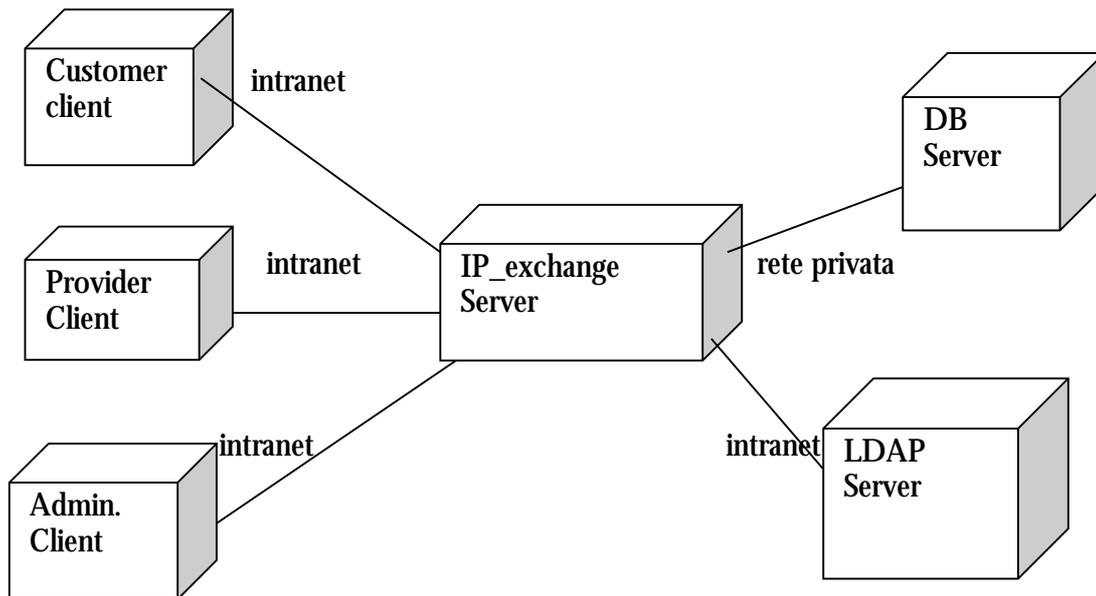


Figura 5.27 Struttura dell'applicazione IPExchange

7.1 Progettazione dettagliata delle classi

Nella fase d'analisi e design sono stati prodotti diversi diagrammi: concettuale, di navigazione e di presentazione. In questi diagrammi è stato introdotto un insieme di classi. In quella fase vengono identificate: le classi, qualche attributo, e, a volte, anche qualche operazione.

Nelle fasi successive, le classi sono arricchite da qualche dettaglio. Quest'attività include le seguenti sotto attività: definire le operazioni, definire gli attributi, identificare le aggregazioni, le associazioni, ereditarietà e dipendenze fra classi, descrivere i metodi, determinare gli stati e stabilire i requisiti rilevanti per la loro implementazione.

Per dare una descrizione dettagliata delle classi introdotte, è necessario utilizzare il *template* presentato nel seguente sottoparagrafo, in esso verranno descritte solo alcune classi del diagramma concettuale.

La descrizione di tutte le classi esula dagli scopi di questa tesi. Questo tipo di documentazione sarà prodotta in seguito, per rilasciare alla STMicroelectronics, la documentazione completa dell'intero prodotto realizzato.

7.2 Descrizione delle classi rilevanti per l'implementazione

Name: CATALOGUES.

Description: elenca tutti gli IP forniti da una determinata divisione.

Inherits from class: none.

Attributes: division, ip_type, technology.

Operations: view(division, ip_type, technology).

Relationships: IP_EXCHANGE.

Diagrams: diagramma concettuale, diagramma della struttura di navigazione, diagramma dello spazio di navigazione, diagramma di presentazione.

Special requirements: None.

Trace: Author.

Name: Division Provider.

Description: definisce le divisioni che forniscono IP.

Inherits from class: none.

Attributes: name.

Operations: choose(), view(), insert(), modify(), delete().

Relationships: People, Manager, Typeof Ip, Request IP.

Diagrams: diagramma concettuale.

Special requirements: None.

Name: TypeOfIP.

Description: definisce il tipo di IP fornito da una determinata divisione.

Inherits from class: none.

Attributes: name.

Operations: view(division, ip_type, technology).

Relationships: Division_Provider, Provider, Request IP.

Diagrams: diagramma concettuale.

Special requirements: None.

Trace: Author.

Name: People.

Description: contiene tutti i dati necessari per individuare una persona che lavora per una divisione provider.

Inherits from class: none.

Attributes: id, name, surname, phone, location, group, mail.
Operations: insert(), view(), modify(), delete().
Relationships: Division_Provider.
Diagrams: Diagramma Concettuale.
Special requirements: None.
Trace: Author.

Name: Manager
Description: contiene tutti i dati necessari per individuare il manager di una determinata divisione provider.
Inherits from class: People.
Attributes: id, name, surname, phone, location, group, mail.
Operations: change_role().
Relationships: Division_Provider.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: Provider
Description: contiene tutti i dati necessari per individuare il provider di uno o più tipi di IP forniti da una determinata divisione provider.
Inherits from class: People.
Attributes: id, name, surname, phone, location, group, mail.
Operations: change_role().
Relationships: TypeOfIP.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: Designer
Description: contiene tutti i dati necessari per individuare i designer di una determinata divisione provider.
Inherits from class: People
Attributes: id, name, surname, phone, location, group, mail.
Operations: change_role(), choose_for_proposal ().
Relationships: Proposal.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: REQUEST IP
Description: contiene tutti i dati necessari per richiedere un IP.
Inherits from class: none.
Attributes: id_ip , ip_name, ip_functionality, release_type, functional_change, other_change, milestones_definition, milestones_intermediate, milestones_full_function,

milestones_final, product_integrated, tape_ship, status, day_request, notes.
Operations: new_request(), view(), change_status(), delete().
Relationships: Customer, Division_Provider, TypeOfIP, Notify, Proposal, Questionnaire, DatabaseManagementSystem, Technology.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: PROPOSAL.
Description: contiene tutti i dati necessari per rappresentare la proposta inoltrata dal provider.
Inherits from class: none.
Attributes: ip_id, ip_name, ip_functionality, release_type, functional_change, other_change, milestones_definition, milestones_intermediate, milestones_fullfunction, milestones_final, status, cost, risk factor, development site, start_work, end_work, day_acceptance, criteria, day_proposal, notes.
Operations: new_proposal(), modify(), view(), change_status(), delete().
Relationships: Notify, DatabaseManagementSystem, Designer, Release.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: NOTIFY.
Description: definisce la notifica mandata dopo determinate azioni.
Inherits from class: none.
Attributes: from, to, cc, bcc, subject, body.
Operations: send() .
Relationships: REQUEST IP, Proposal, Questionnaire.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: Questionnaire.
Description: rappresenta il questionario (una serie di domande) che il customer deve compilare quando la procedura di scambio è stata completata.
Inherits from class: none.
Attributes: question.
Operations: send(), fill(), view().
Relationships: Notify, REQUEST IP, Customer.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: Release
Description: rappresenta l'IP vero e proprio inviato al Customer.
Inherits from class: none.

Attributes: version.
Operations:
Relationships: Proposal.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

Name: Customer
Description: contiene tutti i dati necessari per individuare un Customer.
Inherits from class: User.
Attributes: ID, name, surname, phone, location, group, division, mail.
Operations: insert(), view().
Relationships: Request IP, Questionnaire, Proposal.
Diagrams: diagramma concettuale.
Special requirements: None.
Trace: Author.

7.3 Definizione di sottosistemi e loro interfacce

L'obiettivo, della suddivisione del sistema in sottosistemi, è ottenere un insieme di sottosistemi il più possibile indipendenti fra di loro, in modo tale da essere sviluppati da diversi sviluppatori.

In questo caso, lo sviluppatore è unico, ma la suddivisione in sottosistemi è in ogni caso utile, perché lo sviluppo è più facile e preciso.

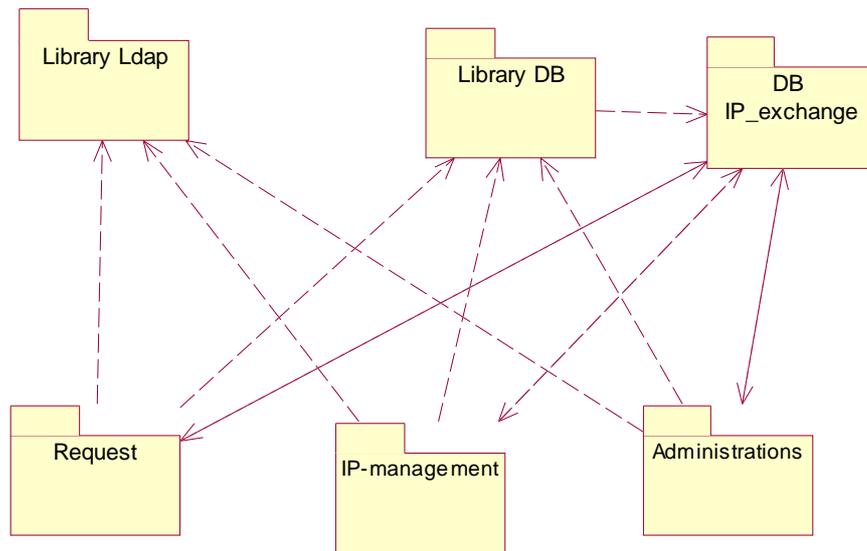


Figura 5.28 Sottosistemi di IP Exchange

8 Implementazione

La fase d'implementazione consiste nella trasformazione dei risultati prodotti nella fase di design (classi, sottosistemi ed interfacce, ecc), in un sistema implementato in termini di componenti (codice, script, eseguibili, ecc.).

Come detto nel capitolo 4, per l'implementazione si utilizza il linguaggio script PHP4, per gli script lato server, mentre come RDBMS si usa mysql.

La struttura dei componenti prodotti rispecchierà quella dei sottosistemi visti in figura 5.28. Infatti, come si può notare in figura 5.29, i componenti del package cmg_des¹⁰ sono stati raggruppati in package (rappresentano le directory) che rispecchiano i sottosistemi visti prima.

8.1 Diagramma dei Componenti

Lo script ipexch.php è lo script principale. Esso implementa la struttura principale dell'applicazione ed in particolare le classi di presentazione della vista del customer, ed inoltre, autentica il customer.

Il database IP-exchange è il database usato per la nostra applicazione. Esso è costituito da un insieme di tabelle, fra loro relazionate. Queste tabelle corrispondono alle classi del diagramma concettuale, le relazioni presenti in quel diagramma sono garantite tramite le relazioni fra le tabelle.

Il package IP DB contiene dei file che saranno inclusi negli script. Essi contengono configurazioni e procedure necessarie per interagire con il Database.

¹⁰ Il package cioè il directory che conterrà i file d'implementazione, si chiama cmg_des e non IP_Exchange perché i file prodotti sono stati inseriti in una directory già esistente per integrare dei file già esistenti, che fanno parte della pagina Web relativa alla divisione CMGDesign.

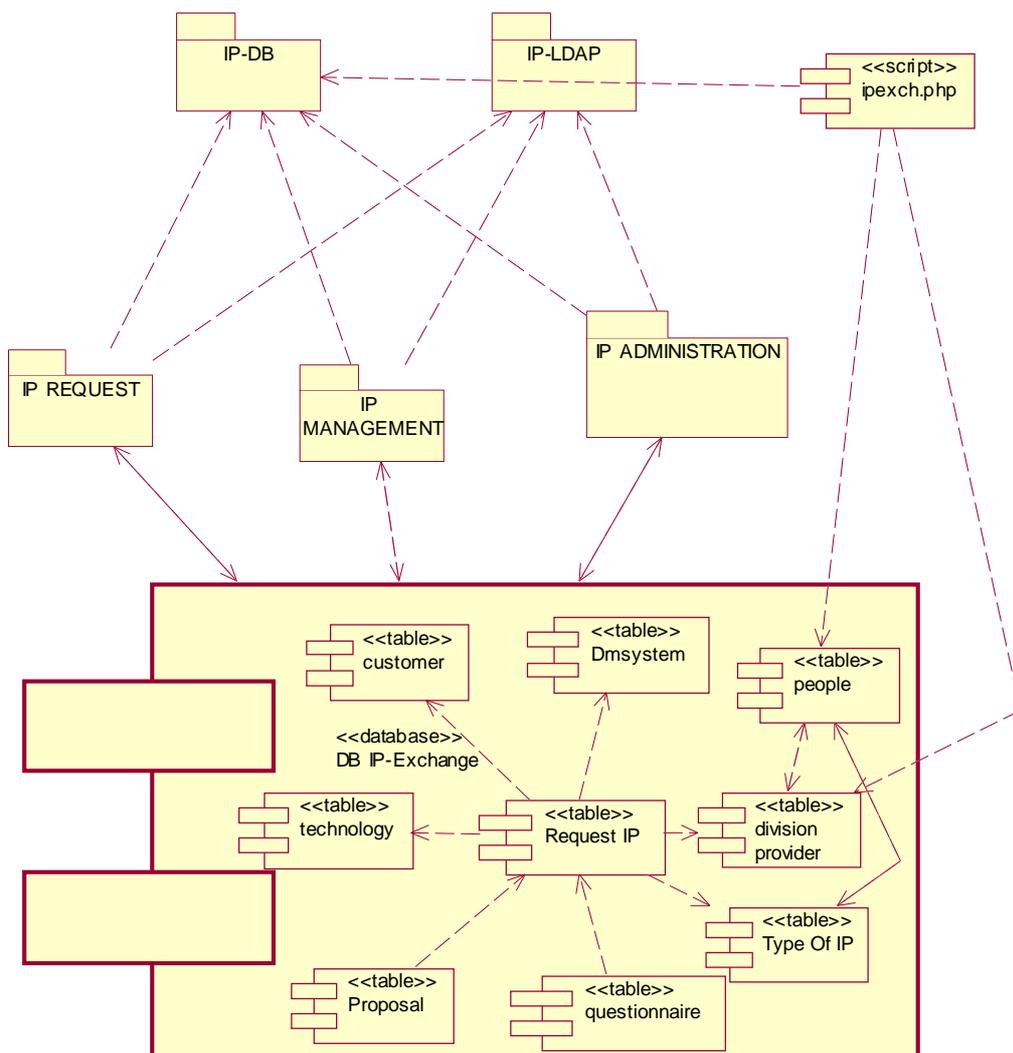


Figura 5.29 Struttura del package cmg_des e del database associato.

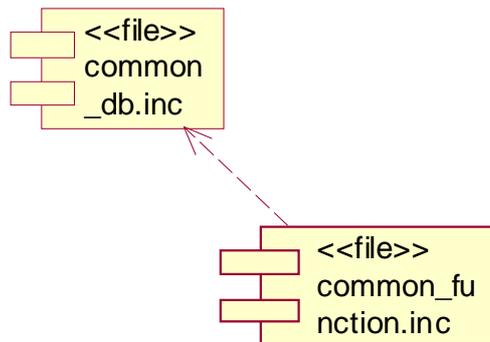


Figura 5.30 Contenuto del package IP DB

Il package IP LDAP contiene dei file che saranno inclusi negli script. Essi contengono configurazioni e procedure necessarie per interagire con il server ldap e recuperare i dati dall'Enterprise Directory.

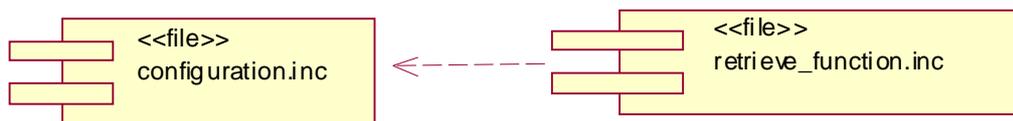


Figura 5.31 Contenuto del package IP LDAP

Il package IP Request contiene gli script che rendono possibile l'invio della richiesta da parte del customer. Tramite essi si può accedere al catalogo (classe CATALOGUE), memorizzare i dati del customer che invia la richiesta (classe CUSTOMER) eseguire il metodo send-request() della classe Customer. Questo metodo, implica l'invio della notifica (classe NOTIFY).

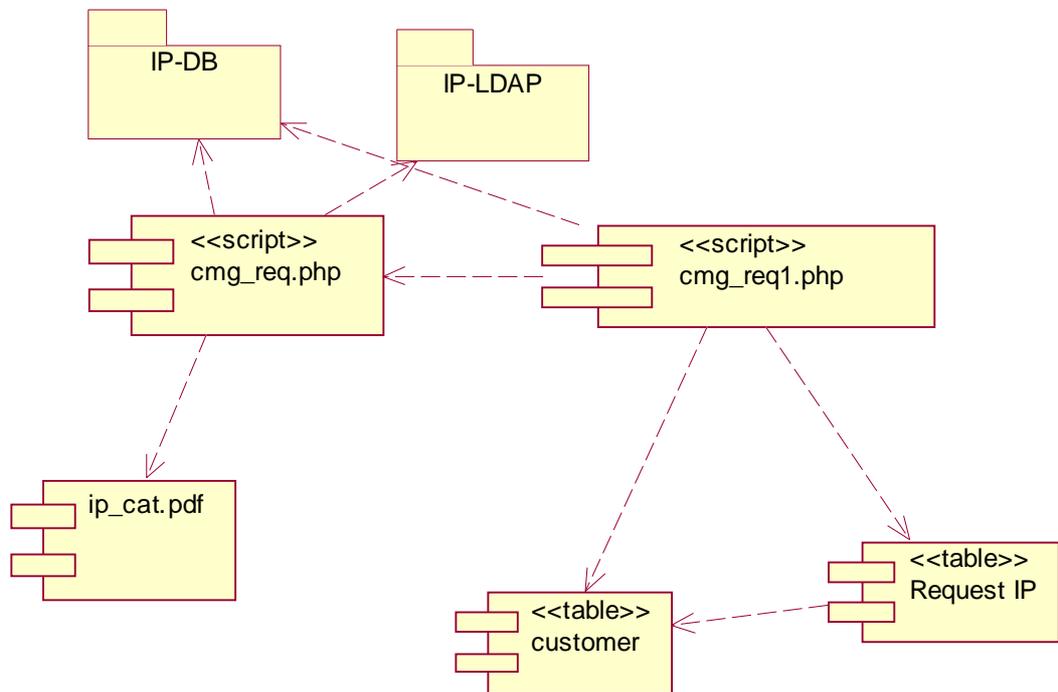


Figura 5.32 Contenuto del package IP request

Il Package IP Administration contiene gli script relativi alla parte amministrativa di IP Exchange (visibili nelle figure 5.33, 5.34). Essi implementano tutte le classi di presentazione relative alla vista relativa all'administrator(tranne il menu che viene implementato tramite lo script act_list.php presente in figura 5.40), Implementa, inoltre, metodi (insert(), update(), delete()), relativi alle classi concettuali "Division Provider", "Type of IP", "Technology", "People".

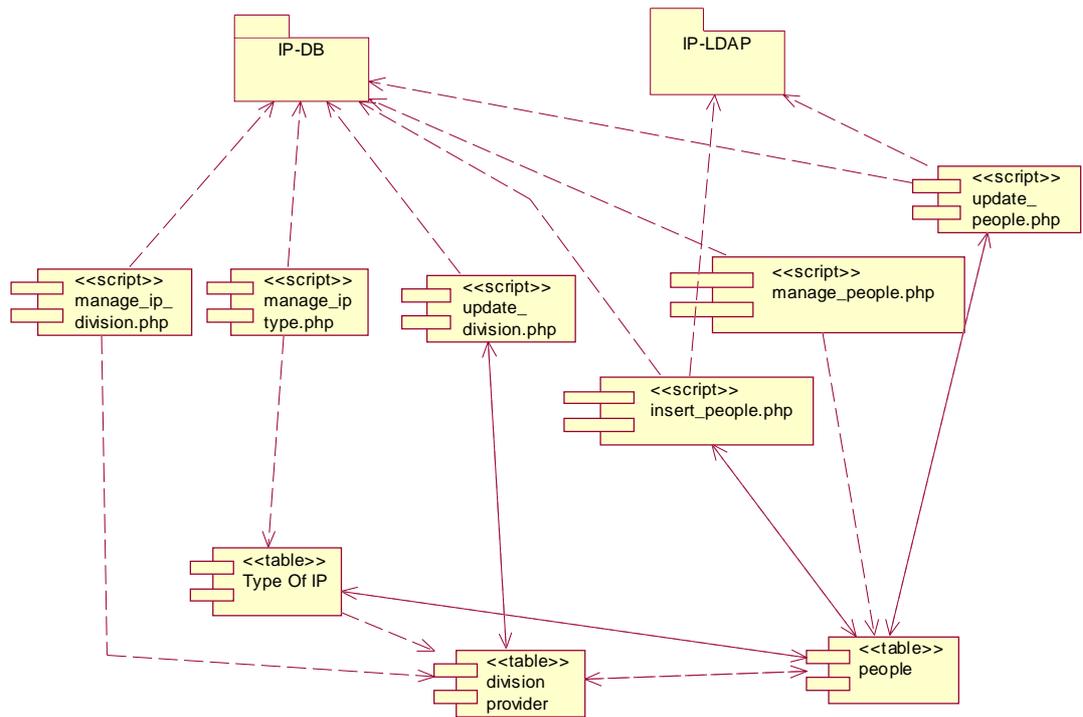


Figura 5.33 Contenido del package IP_Administration

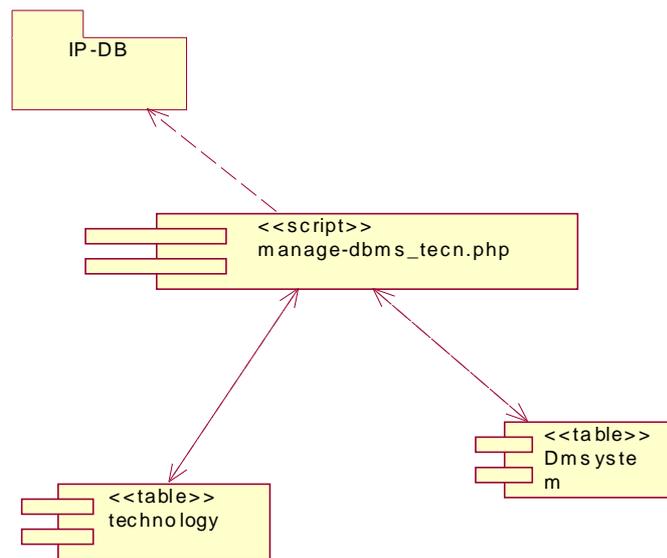


Figura 5.34 Contenido del package IP_Administration

Le figure 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, raffigurano il contenuto del package IP Management. In particolare in figura 5.35 sono visibili tutti gli script necessari per l'invio della proposta; in figura 5.36 sono rappresentati, invece, quelli necessari per cambiare lo stato della richiesta.

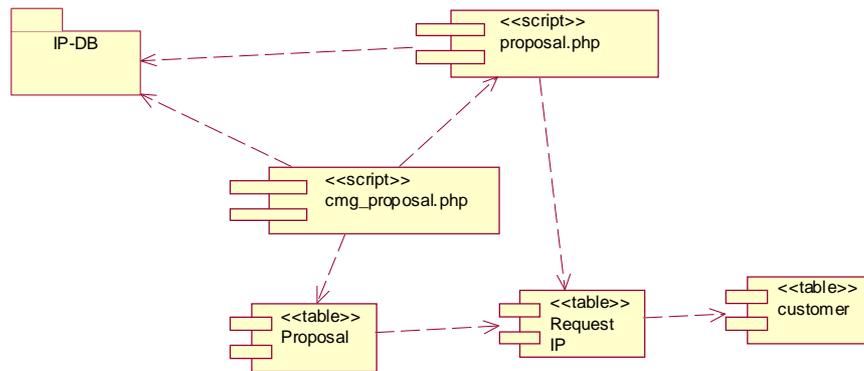


Figura 5.35 Contenuto del package IP_management

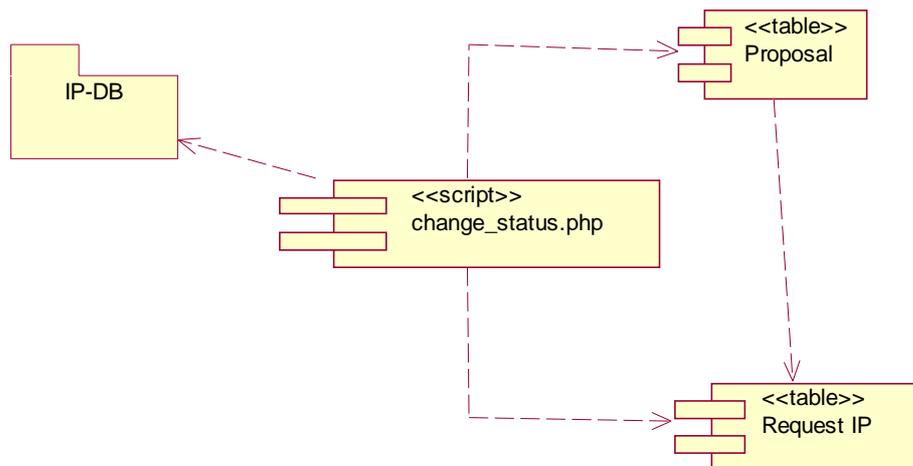


Figura 5.36 Contenuto del package IP_management

In figura 5.37 sono presenti gli script che rendono possibile la cancellazione di "REQUEST IP", "PROPOSAL" e "questionnaire", e la gestione delle richieste ancora attive e di quelle già concluse.

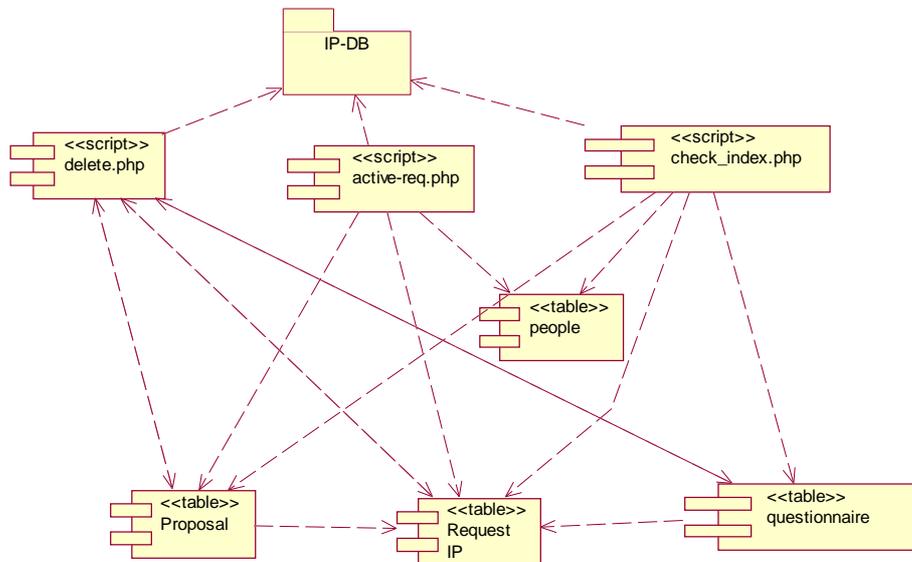


Figura 5.37 Contenuto del package IP_management

In figura 5.38 sono raffigurati gli script che rendono possibile l'invio, la visione e la compilazione del "questionnaire", e la visione della proposta relativa ad una richiesta.

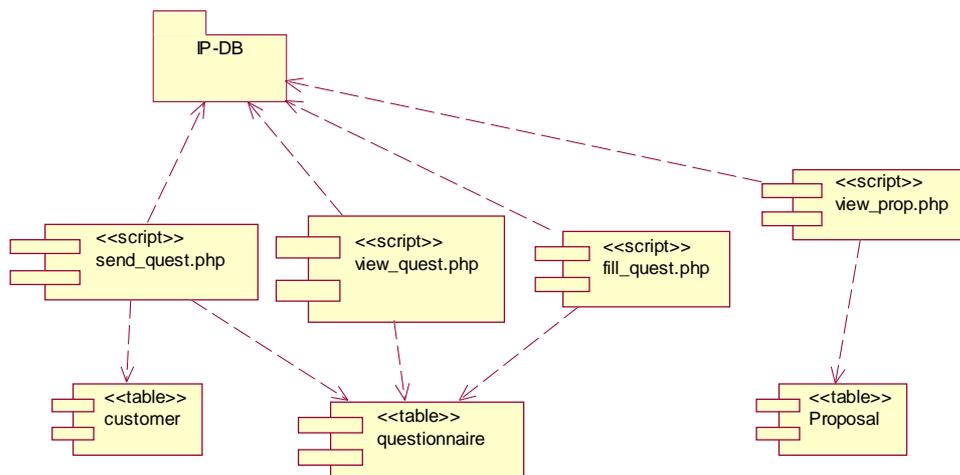


Figura 5.38 Contenuto del package IP_management

In figura 5.39 sono visibili: lo script che permette di visualizzare la classe di presentazione per “REQUEST IP”, e quello che permette l'esecuzione del metodo log_request() della classe provider.

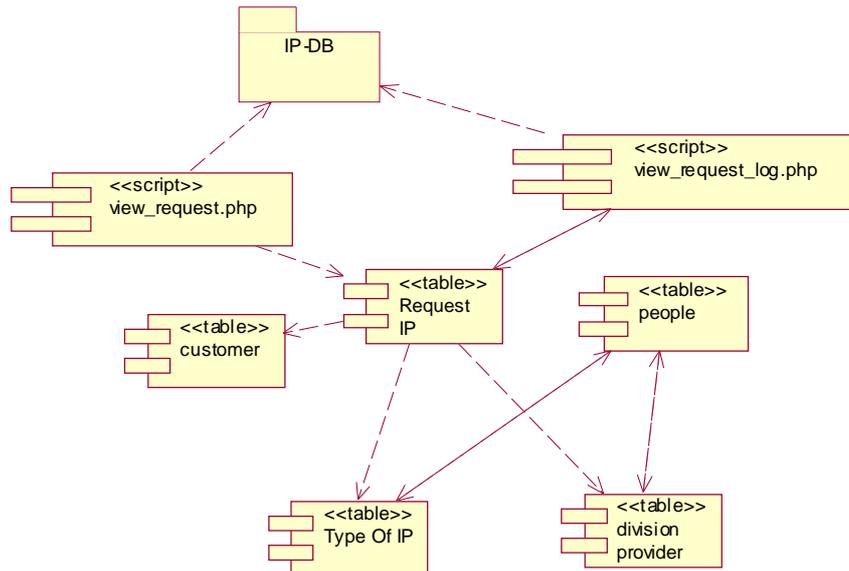


Figura 5.39 Contenuto del package IP_management

In figura 5.40 è presente lo script, tramite il quale vengono autenticati provider ed administrator, e che in seguito all'autenticazione mostra il relativo menu, rispettivamente “Manage IP Main Menu” o “Administration Menu”.

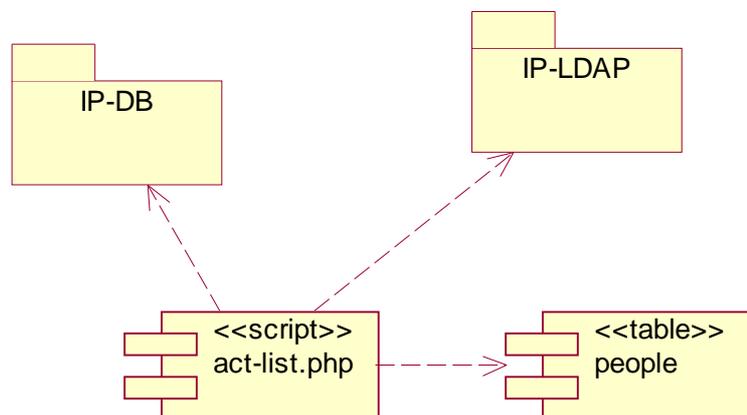


Figura 5.40 Contenuto del package IP_management

8.2 Diagramma di Dispiegamento

I diagrammi di dispiegamento o *deployment diagram* mostrano l'architettura hardware e software del sistema. Nel diagramma in figura 5.41 si può notare come sono interconnessi fra di loro gli elementi fisici dell'applicazione in questione.

I componenti analizzati precedentemente saranno localizzati come segue:

- Il database DB IP Exchange visto in figura 5.29 è localizzato fisicamente nel database server.
- Tutti gli altri package visti sono presenti fisicamente nel Web server all'interno del package cmg_des.
- L'Enterprise Directory si trova fisicamente nel ldap server.

Il client otterrà tutte le informazioni elaborate dal Web server tramite il protocollo HTTP. Mentre fra il Web server ed il database server il protocollo di comunicazione sarà il TCP/IP. Fra Web server e ldap server il protocollo di comunicazione sarà proprio ldap.

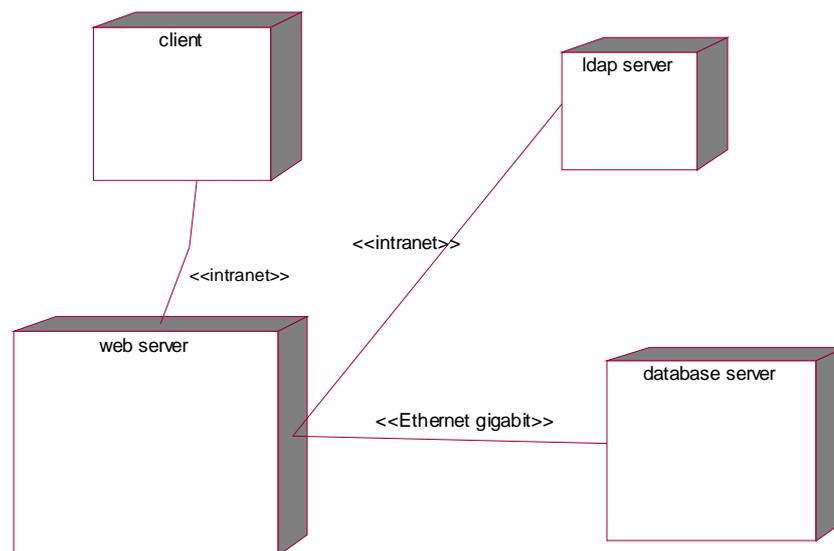


Figura 5.41 Diagramma di dislocamento

Capitolo 6

IL CASO DI STUDIO DVD-STB WEB

DVD-STB Web¹¹ è un'applicazione Web nata per soddisfare le richieste delle divisioni DVD e STB del gruppo CMG della STMicroelectronics di Catania. Lo scopo principale di quest'applicazione è rendere visibili le attività svolte dalle divisioni in questione all'interno della STMicroelectronics, e facilitare lo scambio delle informazioni (*cross fertilization, knowledge sharing*) sia internamente alle divisioni che fra altre divisioni dell'azienda.

1 Requisiti funzionali

I requisiti funzionali di quest'applicazione sono nati in seguito ad una serie d'interviste fatte ai componenti del gruppo, per indagare sulle loro aspettative riguardo l'applicazione.

Dopo questa serie d'interviste, si è compreso che le principali funzioni richieste sono:

1. Scambiare informazioni, sia all'interno che all'esterno della divisione (*cross fertilization, knowledge sharing*).
2. Facilitare la comunicazione tra i componenti dei *team* distribuiti su diversi siti.
3. Pubblicare documenti di vario genere, ad esempio materiale tecnico, specifiche di progetto, *script, report* dei *meeting* ecc.
4. Segnalare un evento importante.
5. Controllare in modo automatizzato il piano delle attività e delle allocazioni dei componenti dei vari team.
6. Aumentare la visibilità della divisione, il lavoro svolto (progetti, prodotti, metodologie e *design*) all'interno della STMicroelectronics.

Le caratteristiche richieste per l'applicazione sono:

1. Autenticare ed autorizzare l'utente.
2. Gestione dell'applicazione facile, dinamica, e trasparente.
3. Aggiornamento automatico dei dati.
4. Riutilizzo di risorse e documenti già esistenti (non duplicazione).

¹¹ Digital Versatile Disk and Set Top Box divisions on the Web

Fra tutte queste caratteristiche, quella maggiormente sentita è la non duplicazione di risorse esistenti, richiesta anche per soddisfare la standardizzazione dei dati interni. Si vuole, quindi, adattare alle esigenze dei componenti del gruppo e delle persone interessate al loro lavoro, le risorse ed i dati già esistenti all'interno della STMicroelectronics. Questo perché, le informazioni, i documenti ed il modo di consultarli esiste già, il problema è che esistono troppe informazioni, ed altrettanti strumenti utilizzabili per il loro recupero. Quindi l'intervento della nostra applicazione mira proprio a selezionare le informazioni rilevanti, ed a presentarle in modo consistente.

Quest'esigenza nasce soprattutto per quanto riguarda la gestione dei dati di persone o dei documenti tecnici che si vogliono pubblicare o a cui si vuole accedere.

In entrambi i casi non si vuole gestire i dati localmente, perché ciò comporterebbe la necessità di risorse per l'aggiornamento dei dati (dipendenti, tempo sottratto al lavoro) che non sono disponibili.

Il lavoro richiesto per quanto riguarda questa parte, è stato quello di analizzare le applicazioni esistenti che trattassero le informazioni viste sopra, e quindi sceglierne quella migliore e facilmente integrabile con gli altri requisiti dell'applicazione "DVD-STB Web".

Come già detto nel capitolo 4, dopo un'attenta analisi sono emerse tre applicazioni che verranno integrate a "DVD-STB Web":

1. Div-Web;
2. Enterprise Directory ;
3. The Group Service.

DIV_Web è stato scelto per la gestione dei documenti. E' un prodotto CMG, che gestisce documenti riguardanti altre divisioni CMG, quali MCD e MCDT. E' un prodotto affidabile, versatile, personalizzabile secondo le esigenze delle diverse divisioni, e soprattutto consente di gestire i documenti in modo centralizzato, facile e con poche operazioni. Risponde ai

requisiti di sicurezza e alta disponibilità (High Availability), che sono gli *standard* di qualità applicati da DVD e STB a tutto il flusso di progettazione: Capability Maturity Model (CMM) per il software e ISO 9001 per l'hardware ed il software.

Per la gestione dei dati riguardanti le persone il compito è stato più difficile, perché queste informazioni verranno trattate in due modi differenti:

1. Autenticare ed autorizzare chi accede a determinate informazioni;
2. Trattare le informazioni riguardanti gruppi di persone.

Per quanto riguarda il primo punto, la scelta dell'applicazione da utilizzare è stata quasi obbligata, perché l'**Enterprise Directory** gestisce in modo centralizzato tutti i dati riguardanti tutti i dipendenti ST e gli eventuali accessi che essi hanno ai vari servizi interni.

Per quanto riguarda il secondo punto, il compito è stato un po' più difficile, perché si voleva creare uno strumento che aggiornasse, in modo più automatico e trasparente possibile, i dati di un team di persone. Ma dopo una serie di ricerche si è appreso che per politiche aziendali la gestione di gruppi di persone non è concesso, se non attraverso l'applicazione **The Group Service**, che è stata analizzata nel capitolo 4.

2 Requisiti non funzionali

La struttura dell'hardware e del software disponibile è stata descritta nel capitolo 4.

3 Funzionalità che DVD_WEB offre agli utente

1. Bachecca elettronica;
2. La possibilità di visualizzare e gestire i report degli "operation meeting";
3. Gestione dei progetti sviluppati dalla divisione DVD:
 - a) Creazione di un nuovo progetto (documentazione e team);
 - b) Modifica dei dati dei progetti esistenti.
4. Gestione delle attività svolte dai vari gruppi funzionali che compongono la divisione DVD. In particolare si vedranno i dati dei seguenti gruppi funzionali:
 - a) EDA.
 - b) Timing Analysis.

- c) Training.
 - d) Analog.
5. Per il gruppo funzionale EDA le funzionalità offerte sono:
- a) Visualizzazione del team.
 - b) Modifica team.
 - c) Gestione e visualizzazione dei dati riguardanti i tool interni e commerciali utilizzati dall'intera divisione DVD.
 - d) Gestione della scadenza delle licenze dei tool commerciali.
 - e) Visualizzazione delle informazioni che riguardano i *design environment* mantenuti e assistiti da EDA.
 - f) Gestione dell'utilizzo dei *tool* da parte degli altri gruppi funzionali.
6. Per quanto riguarda il gruppo funzionale Timing Analysis, le funzionalità offerte sono:
- a) Visualizzazione del team.
 - b) Modifica team.
 - c) Visualizzazione dei manuali d'uso riguardanti i tool EDA interni e commerciali utilizzati dal gruppo funzionale Timing Analysis, ma gestiti dal gruppo funzionale EDA..
 - d) Gestione dei documenti e metodologie riguardanti i signoff.
 - e) Gestione degli *script*.
7. Per il gruppo funzionale Analog le funzionalità offerte sono:
- a) Visualizzazione del team.
 - b) Modifica team.
 - c) Visualizzazione dei manuali d'uso riguardanti i tool EDA interni e commerciali utilizzati dal gruppo funzionale Analog, ma gestiti dal gruppo funzionale EDA..
 - d) Gestione delle informazioni riguardanti i *design*.
 - e) Gestione delle informazioni riguardanti i progetti e signoff.
8. Per quanto riguarda il gruppo funzionale Training le funzionalità offerte sono:
- a) Visualizzazione del team.
 - b) Modifica team.
 - c) Visualizzazione del un catalogo dei corsi ed informazioni utili divise per area d'interesse.
 - d) Gestione dei provider dei corsi.
 - e) Gestione delle training form.

4 Casi d'uso

Nel paragrafo seguente saranno descritti i casi d'uso in modo molto dettagliato, utilizzando gli scenari primari e secondari ed i diagrammi associati.

Sia per la loro quantità che per la loro struttura è preferibile raggruppare i casi d'uso in un insieme di viste; ovvero i casi d'uso sono stati divisi in “package”. In figura possiamo vedere i package che compongono i casi d'uso.

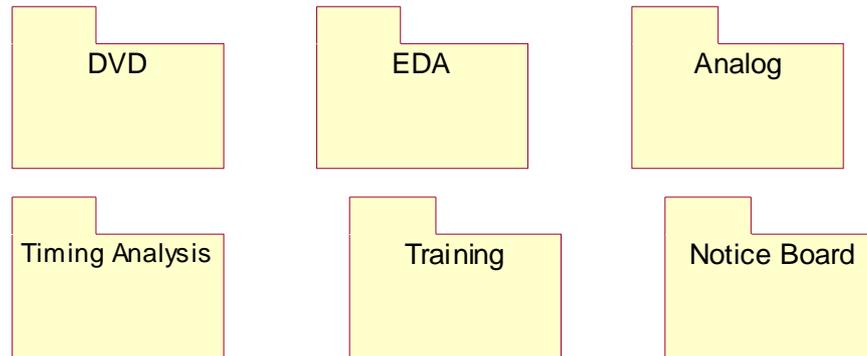


Figura 6.1. Package per i casi d'uso

In Particolare: in figura 6.2 si può notare il diagramma dei casi d'uso per il package DVD, mentre in figura 6.3 è raffigurata la vista riguardante il package EDA, in figura 6.4 è raffigurata la vista riguardante il package Timing Analysis, in figura 6.5 è raffigurata la vista riguardante il package Analog, in figura 6.6 è raffigurata la vista riguardante il package Training, ed in figura 6.7 è raffigurata la vista riguardante il package Notice Board.

4.1 Attori

Gli attori nella descrizione dei requisiti sono: STPeople, TeamLeader, cronjob, CMGPeople.

Actor: STPeople.
Description: Uno STPeople è un dipendente ST non autenticato.
Use Case DVD_CT_operation_Meeting. View All Project. Notice Board. Functional Group. Team EDA. Tool Commercial And Internal. Documentation About Design. Team Timing Analysis. User Guide Tool used by Timing Analysis. Script. SignOff.

Team Analog.
Technical docs.
Design.
Project By Analog.
User Guide Tool used by Analog.
Team Training.
Training And Information.
Course Provider.
Training Form
View Notice ST.

Actor: CMGPeople
Description: Un CMGPeople è un dipendente ST autenticato, che lavora per il gruppo CMG Catania.
Use Case
View notice.
Insert new notice.

Actor: TeamLeader
Description: Un TeamLeader è il responsabile di un gruppo di persone. E' quindi autenticato.
Use Case
Project.
New Project.
Update Project.
Administration.
Insert New Documentation.
Insert New Team.

Actor: Cronjob
Description: Un Cron job è processo che esegue una determinata azione automaticamente.
Use Case
Licence expiration for commercial tool.
Send notify of expiration notice to owner.
CleanUP.
Update All Team.

Actor: Owner
Description: Un Owner è un CMGPeople che ha inserito un evento nella Notice Board.
Use Case
Update Notice.
Modify Notice.
Delete Notice.
Confirm/refuse Expiration.

4.2 Casi d'uso in dettaglio

La descrizione dei casi d'uso inizia con quelli appartenenti al package DVD rappresentati in figura 6.2.

Use case name: "DVD_CT Operation Meeting".

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: La pagina Web dvd.ctn.st.com è visualizzata ed è stato selezionato il link "dvd_ct_operation meeting".

Post-condition: Il report selezionato è visualizzato.

Use case "included": "Recover documentation".

Flow of events:

1. Questi report sono contenuti nel database dell'applicazione "DIV-Web", essi sono caratterizzati da category="management" e doctype="staff meeting minutes".
2. Per recuperarli il controllo passa al caso d'uso "recover documentation", al quale verranno passati le seguenti informazioni: category="management" e doctype="staff meeting minutes".
3. I record recuperati, saranno ordinati secondo la data d'inserzione in modo decrescente, e raggruppati per anno.
4. Verranno visualizzati quindi tanti link quanti sono i record estratti, tramite i quali l'utente potrà selezionare il report desiderato.

Use case name: "Insert new documentation".

Actors: TeamLeader.

Priority: 1.

Status: Requirements capture.

Flow of events:

1. Per inserire documentazione il controllo passa all'applicazione "DIV_Web".

Use case name: "Insert new Team".

Actors: TeamLeader.

Priority: 1.

Status: Requirements capture.

Flow of events:

1. Per inserire un nuovo team il controllo passa all'applicazione "The Group Service".

Use case name: “Functional Group”.
Actors: STPeople.
Priority: 1.
Status: Requirements capture.
Pre-condition: L’utente seleziona il link functional group.
Post-condition: L’utente può, scegliere di visitare la pagina Web del gruppo funzionale desiderato.
Flow of events:
1. Il caso d’uso inizia quando l’utente seleziona il link “functional group”.
2. Verranno visualizzati tanti link quanti sono i gruppi funzionali per i quali è stata creata una pagina Web: “EDA”, “Analog”, “Training”, “Timing Anlysis”.
3. I link punteranno alle relative pagine Web.
4. L’utente può scegliere quale gruppo funzionale andare a visitare.

Use case name: “Project”.
Actors: TeamLeader, STPeople.
Priority: 1.
Status: Requirements capture.
Pre-condition: La pagina Web dvd.ctn.st.com è visualizzata ed è stato selezionato il link “Project”.
Post-condition: Le informazioni che riguardano i progetti prodotti da DVD e STB sono visualizzate.
Use case “included”: “View all Project”, “Update project”, “New project”.
Flow of events:
1. In automatico è attivato il caso d’uso “View all Project”.
2. E’ data la possibilità di scegliere se creare un nuovo progetto o modificarne uno esistente.
3. Se si sceglie di creare un nuovo progetto il controllo passa al caso d’uso “New Project”.
4. Mentre se si sceglie di modificarne uno esistente il controllo passa al caso d’uso “Update Project”.

Use case name: “Authenticate ”.
Actors: Team Leader.
Priority: 1.
Status: Requirements capture.
Pre-condition: L’utente seleziona il link relativo all’ area amministrativa, ove è concesso l’accesso solo ad un team leader.
Post-condition: Il TeamLeader è stato riconosciuto dal sistema.
Flow of events:
1. Il caso d’uso inizia quando è visualizzata la finestra che consente l’inserimento di login e password.

2. In questa finestra ci sono due bottoni: “ok” e “cancel”.
 3. Se l'utente seleziona il bottone “ok”:
 - a) Il sistema si collega al server ldap;
 - b) Si ricerca un record che contenga la Login fornita dall'utente, nella “Enterprise Directory”;
 - c) Si verifica se la Password inserita coincide con quella memorizzata nel record appena prelevato.
 - d) Si controlla se l'utente appena autenticato è riconosciuto come teamleader di un qualche gruppo.
 - e) Se è presente allora l'attore sarà riconosciuto come TeamLeader, quindi potrà accedere all' area amministrativa.
- Secondary scenario:**
- A. 3. b) Il server ldap non è disponibile
c) L'utente non si potrà autenticare.
 - B. 3. b) La login fornita è sbagliata
c) L'utente non sarà autenticato deve riprovare di nuovo ad inserire una login giusta.
 - C. 3. b) La Password fornita è sbagliata
c) L'utente non sarà autenticato deve riprovare di nuovo ad autenticarsi.

- Use case name:** “New Project”.
- Actors:** TeamLeader.
- Priority:** 1.
- Status:** Requirements capture.
- Pre-condition:** Il team leader è autenticato, si trova nell'area amministrativa dell'applicazione e seleziona il link “New Project”.
- Post-condition:** Le informazioni che caratterizzano un progetto sono state inserite.
- Use case “included”:** “Insert new documentation”, “Insert new Team”.
- Flow of events:**
1. L'utente viene identificato ed autenticato tramite “autenticare leader”, solo il team leader può creare un nuovo progetto.
 2. Le azioni da fare per creare un nuovo progetto sono: “creare il team”, ed “inserire la documentazione relativa”.
 3. Tramite un form, si può memorizzare localmente il nome del team, l'indirizzo e-mail del team leader relativo, il Project file ed una breve descrizione per progetto.
 4. Se l'inserimento è avvenuto, il controllo passa a “DIV_Web” per l'inserimento della documentazione relativa.
 5. Mentre per creare il nuovo team il controllo passa a “The Group Service”.

Use case name: “Update Project”.

Actors: TeamLeader.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il team leader è autenticato, si trova nell’area amministrativa dell’applicazione e seleziona il link “Update Project”.

Post-condition: Le informazioni che caratterizzano un progetto sono state modificate.

Flow of events:

1. L’utente viene identificato ed autenticato tramite “autenticare leader”, solo il team leader può modificare un progetto.
2. Le informazioni relative ai progetti esistenti sono adesso visualizzate.
3. Il team leader sceglie un progetto da modificare.
4. Sarà visualizzato un form dal quale si possono modificare il Project file, la descrizione del progetto, il nome del team relativo o l’indirizzo e-mail del team leader.
5. Su questo form saranno presenti due bottoni tramite i quali si può confermare la modifica o meno.
6. Se non si conferma la modifica tutto rimane invariato.

Use case name: “View Team”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il nome del gruppo da visualizzare deve essere noto.

Post-condition: Le informazioni riguardanti il gruppo in questione sono visualizzate.

Use case “included”: “Recover Team”.

1. Il gruppo è già stato recuperato, quindi le informazioni riguardanti il gruppo sono reperibili localmente.
2. Le informazioni vengono recuperate e visualizzate.

Secondary scenario:

1. Il gruppo non è stato recuperato.
2. Il controllo passa al caso d’uso “recover team”.

Use case name: “View all Project”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link “Project” della pagina dvd.ctn.st.com è stato selezionato.

Post-condition: Le informazioni riguardanti i progetti attivi sono adesso visualizzate.

Use case “included”: “view Team”, ”recover documentation”.

Flow of events:

1. I progetti sono identificati univocamente tramite il documento “Project file” presente nel database di “Div-Web”, avente come category: ”Project” e doctype: ”Project file”.
2. I “Project file” ed i nomi dei “Team” associati, dei progetti prodotti da DVD-STB Catania sono memorizzati localmente.
3. Per recuperare tutti i documenti riguardanti un progetto e le varie fasi di costruzione di esso si deve fare una ricerca nel database, essi saranno tutti quei documenti che sono caratterizzati dalla presenza di un identificativo di progetto nel campo “Project file”.
4. Il controllo passa al caso d’uso “recover documentation”, come parametri verranno passati category=’all’, doctype=’all’, Project file=’noto’. I record recuperati vengono adesso ordinati per Project file, category, e doctype e quindi visualizzati.
5. Per ogni progetto visualizzato sarà data la possibilità di visualizzare anche il relativo team. Se l’utente vorrà visualizzare team relativo ad un determinato progetto il controllo passerà al caso d’uso “view Team” al quale verrà passato il nome del gruppo da visualizzare.

Use case name: “recover documentation”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: I campi secondo cui viene fatta la ricerca sono noti.

Post-condition: I documenti desiderati sono adesso disponibili per essere visualizzati.

Flow of events:

1. La gestione della documentazione avviene tramite “DIV-Web”, fisicamente questi documenti si trovano nel database “xx” ed esattamente nella tabella “dvd”, ubicato nella macchina “yy” di Rousset. [Nota: “xx” e “yy” sono dei dati indicativi, verranno sostituiti con i dati reali per la documentazione che verrà rilasciata in azienda].
2. I diversi tipi di documenti sono caratterizzati tramite i seguenti campi: category, doctype e family.
3. I documenti specifici di un determinato progetto sono identificati univocamente tramite

un identificativo memorizzato nel campo “Project file”;

4. Per andare a recuperare tutti i documenti desiderati si deve fare una ricerca nel database dell'applicazione “DIV_Web”, secondo i campi category, doctype, family e project file passati dai casi d'uso chiamanti.
5. I record recuperati vengono adesso ordinati per project file, per family, per category, ed infine per doctype e quindi visualizzati

Use case name: “recover team”.

Actors: Cronjob, STPeople, TeamLeader.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il nome del gruppo da recuperare deve essere noto.

Post-condition: Le informazioni desiderate riguardanti i membri del gruppo sono disponibili per successive ricerche.

Flow of events:

1. I team sono gestiti da “The Group Service”.
2. Per andare a recuperare il team desiderato bisogna collegarsi ad un sever ldap e fare un query per ricercare il gruppo desiderato.
3. Il gruppo è stato trovato, il campo “uniquemember” del record trovato è un campo multivalore nel quale sono memorizzati i “dn” dei membri del gruppo.
4. tramite i “dn” trovati si ricercano le informazioni dei membri del gruppo nella “Enterprise Directory”.
5. Per ogni membro del gruppo verranno estratte le seguenti informazioni:
 - a) Nome e cognome;
 - b) Città;
 - c) Gruppo
 - d) Divisione;
 - e) E-mail;
 - f) Telefono.
6. Le informazioni recuperate saranno memorizzate localmente. Per rendere più veloci gli accessi a questi dati fatti tramite gli altri casi d'uso.

Secondary scenario:

3. Il gruppo desiderato non è stato trovato.
4. Sarà visualizzato un messaggio d'errore.

Use case name: “Administration”.

Actors: TeamLeader.
Priority: 1.
Status: Requirements capture.
Pre-condition: L'utente è autenticato. Si trova nell'area amministrativa dell'applicazione e seleziona "Configuration and team".
Post-condition: Le configurazioni sono state modificate.
Flow of events:
NOTA: per l'esecuzione dell'applicazione devono essere gestite delle informazioni che saranno memorizzate localmente, come i nomi dei team dei gruppi funzionali ed i relativi team leader, il nome della divisione, dei gruppi funzionali, degli standard aziendali per la creazione delle pagine Web.

1. Un form che consentirà l'inserimento delle configurazioni necessarie per l'applicazione è visualizzato.
2. E' possibile inserire, o modificare il nome della divisione.
3. E' possibile inserire, o modificare per ogni gruppo funzionale il nome del team relativo, e l'indirizzo e-mail del teamleader (unico per ogni dipendente ST).
4. Inserire o modificare, gli standard per la costruzione delle pagine (ad esempio il colore dello sfondo).

Use case name: "Update All Team".
Actors: Cronjob.
Priority: 1.
Status: Requirements capture.
Pre-condition: La data e l'ora memorizzate nel cronjob sono scattate.
Post-condition: I team sono stati aggiornati.
Use case "included": "recover team".
Flow of events:

1. I vari team (dei gruppi funzionali o dei progetti) sono gestiti dall'applicazione "The Group Service". Per accedere ai dati relativi bisogna collegarsi al server ldap. Si vuole tenere copia locale dei dati relativi ai team. Periodicamente questi dati vengono aggiornati.
2. Si effettua il collegamento al server relativo e si cercano i gruppi desiderati.
3. Per ogni gruppo si richiama il caso d'uso "recover team". Le informazioni locali vengono aggiornate.

Secondary scenario:

3. Se uno o più gruppi non vengono trovati, o se non è possibile collegarsi al server la situazione rimane invariata. Verrà comunicato al team leader il motivo del mancato

“update” delle informazioni.

Use case name: “Notice_board”.

Actors: CMGpeople, STPeople, Owner.

Priority: 1

Status: Requirements capture.

Pre-condition: Il link relativo alla Notice board è stato selezionato.

Post-condition: La Notice Board è adesso attiva.

Use case “included”: “View notice”, “New Notice”, “ Update notice”, “Confirm/Refuse Expiration”, “Notice expiration notice to owner”, “Authenticate”.

Flow of events:

1. L’utente della Notice-board deve essere autenticato.
2. Il controllo passa al caso d’uso “Authenticate”.
3. L’utente è autenticato.
4. L’utente appartiene alla divisione CMG, il controllo passa al caso d’uso “View notice”.
5. E’ data la possibilità di inserire un nuovo evento tramite il caso d’uso “New Notice”.
6. Se l’utente è un owner avrà anche la possibilità di aggiornare gli eventi da lui inseriti tramite “Update notice”.

Secondary scenario:

- A. 4. L’utente non appartiene alla divisione CMG, il controllo passa al caso d’uso “view notice ST”.
- B. 3. L’utente non è autenticato.
 4. Non potrà accedere alla notice board.

Use case name: “Tool Commercial and internal”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai tool contenuto nella pagina del gruppo funzionale EDA è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata.

Use case “included”: “Recover documentation”.

Flow of events:

1. I tool (commerciali od interni) sono identificati univocamente tramite un identificativo memorizzato nel campo “Tool ID” nei record dei documenti che riguardano i tool.
2. Tutti i documenti riguardanti un tool si trovano sotto la category EDA ed hanno i seguenti doctype: Tool development Plan and report, Tool release notes, Tool Specification, Evaluation spec/report, Procedures. I documenti saranno raggruppati per identificativo di tool presente nel campo “Tool ID”;
3. Il controllo passa al caso d’uso “recover documentation”, come parametri verranno passati category=”EDA”, doctype=”Tool development Plan and report” o “Tool release notes” o “Tool Specification” o “Evaluation spec/report, procedures”, Tool ID=all.
4. I record recuperati vengono adesso ordinati per Tool ID, family (Commercial EDA Tools od Internal EDA Tools), e doctype, e quindi visualizzati.

Use case name: “License expiration for commercial tool”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’utente seleziona il link relativo a license expiration.

Post-condition: Le scadenze dei tool sono visualizzate.

Flow of events:

1. Un cronjob aggiorna periodicamente un file che contiene tutte le scadenze dei tool commerciali usati da EDA.
2. Si accede a questo file.
3. I record recuperati vengono visualizzati.

Use case name: “Documentation About Design”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai Design contenuto nella pagina del gruppo funzionale EDA è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere consultata.

Use case “included”: “recover documentation”.

Flow of events:

1. Per recuperare tutti i documenti riguardanti un design si deve fare una ricerca nel db DIV_Web, per recuperare tutti quei documenti che si trovano sotto la category=EDA ed hanno i seguenti doctype: Design Environment, Methodologies.
2. Il controllo passa al caso d’uso “recover documentation”, come parametri verranno passati category=EDA, doctype=Design Environment, methodologies.
3. I record recuperati vengono adesso ordinati per doctype, e quindi visualizzati.

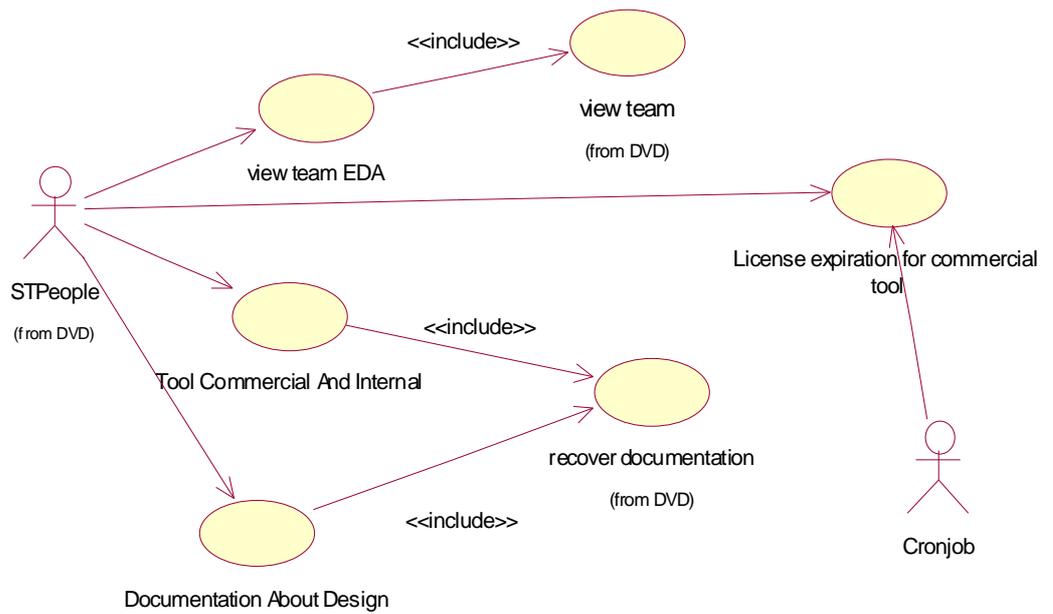


Figura 6.3 Diagramma dei Casi d’uso (Package EDA)

Vediamo, adesso, casi d'uso del package Timing Analysis, i quali sono visibili in figura 6.4.

Use case name: "Team Timing Analysis".

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link "Team Timing Analysis" contenuto nella pagina del gruppo funzionale Timing Analysis è stato selezionato.

Post-condition: L'elenco dei membri appartenenti al team è visualizzato.

Use case "included": "View team".

Flow of events:

1. Si recupera il nome del gruppo che contiene il team Timing Analysis.
2. Il controllo passa al caso d'uso "View team", al quale viene passato anche il nome del gruppo in questione.

Use case name: "script".

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo agli script contenuto nella pagina del gruppo funzionale Timing Analysis è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata.

Use case "included": "recover documentation".

Flow of events:

1. Gli script creati dal team Timing Analysis sono gestiti tramite "DIV-Web".
2. Per recuperarli tutti si deve fare una ricerca nel database, cercando tutti quei documenti che si trovano sotto la category Timing Analysis e come doctype scripts.
3. Il controllo passa al caso d'uso "recover documentation", come parametri verranno passati category=Timing Analysis, doctype=scripts.
4. I record recuperati vengono adesso ordinati per data di inserzione degli script, e quindi visualizzati.

Use case name: "Signoff".

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai Singoff contenuto nella pagina del gruppo funzionale Timing Analysis è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata.

Use case "included": "Recover documentation".

Flow of events:

1. Per recuperare tutti i documenti riguardanti un signoff si deve fare una ricerca nel database dell'applicazione "DIV_Web", per recuperare tutti quei documenti che si trovano sotto la category Timing Analysis ed hanno i seguenti doctype: Signoff criteria, Signoff;
2. Il controllo passa al caso d'uso "recover documentation", come parametri verranno passati category=Timing Analysis, doctype=Signoff criteria, Signoff.
3. I record recuperati vengono adesso ordinati per famiglie e per doctype, e quindi visualizzati.

Use case name: "User Guide of tool used by Timing Analysis".**Actors:** STPeople.**Priority:** 1.**Status:** Requirements capture.**Pre-condition:** Il link relativo ai tool contenuto nella pagina del gruppo funzionale Timing Analysis è stato selezionato.**Post-condition:** La documentazione è adesso disponibile per essere visualizzata.**Use case "included":** "Recover tool EDA user guide".**Flow of events:**

1. I tool sono gestiti dal gruppo funzionale EDA, quindi per quanto riguarda Timing Analysis esso potrà visualizzare solo i manuali d'uso dei tool da esso utilizzati.
2. Il controllo passa al caso d'uso "recover tool user guide" al quale viene passato il nome del gruppo funzionale Timing Analysis.
3. I record trovati verranno raggruppati per famiglie (commercial EDA tools, internal EDA tools), e quindi visualizzati.

Use case name: "recover tool EDA user guide".**Actors:** STPeople.**Priority:** 1.**Status:** Requirements capture.**Pre-condition:** Il valore del campo User secondo il quale si deve fare la ricerca deve essere noto.**Post-condition:** I manuali dei tool desiderati sono stati recuperati.**Flow of events:**

1. I record del database Div-Web che riguardano i tool gestiti da EDA hanno un campo User che indica quali gruppi funzionali useranno questi tool.
2. Per recuperare i documenti desiderati si deve fare una ricerca nel database dell'applicazione "DIV_Web", secondo i campi category="EDA", doctype="Tool User

Guide”, User=”noto all’inizio del caso d’uso” o “ALL”.

3. I record recuperati vengono adesso ordinati per Tool ID, per family (commercial EDA Tools o internal EDA Tools).

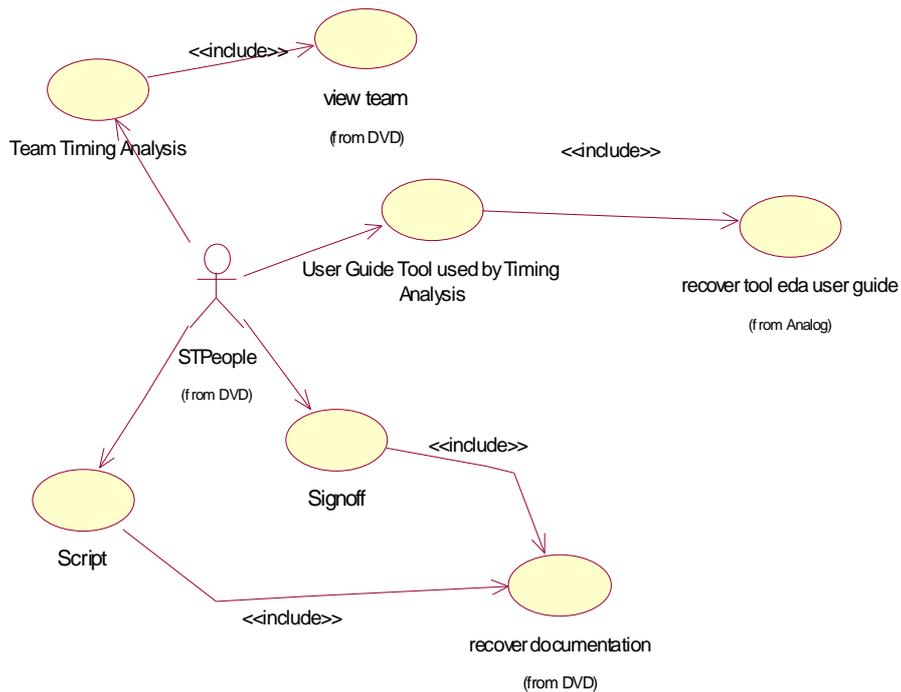


Figura 6.4 Diagramma dei Casi d’uso (Package Timing Analysis)

Di seguito, vengono analizzati i casi d’uso del package Analog, visibili in figura 6.5.

Use case name: “Team ANALOG”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link “Team Analog” contenuto nella pagina del gruppo funzionale Analog è stato selezionato.

Post-condition: L’elenco dei membri appartenenti al team è visualizzato.

Use case “included”: “View Team”.

Flow of events:

1. Si recupera il nome del gruppo che contiene il team ANALOG.
2. Il controllo passa al caso d’uso “view team”, al quale viene passato anche il nome del gruppo in questione.

Use case name: “Technical Docs”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai Technical Docs contenuto nella pagina del gruppo funzionale Analog è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata.

Use case “included”: “recover documentation”.

Flow of events:

1. Per recuperare tutti i documenti riguardanti un Technical Docs si deve fare una ricerca nel database dell'applicazione “DIV_Web”, per recuperare tutti quei documenti che si trovano sotto la category Analog ed hanno doctype Technical docs.
2. Il controllo passa al caso d'uso “recover documentation”, come parametri verranno passati category=Analog, doctype= Technical Docs, Family= “pll”, “clockgen”, “technology”.
3. I record recuperati vengono adesso ordinati per famiglie, e quindi visualizzati.

Use case name: “Desing”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai Design contenuto nella pagina del gruppo funzionale Analog è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata.

Use case “included”: “recover documentation”.

Flow of events:

1. Per recuperare tutti i documenti riguardanti un design si deve fare una ricerca nel database dell'applicazione “DIV_Web”, per recuperare tutti quei documenti che si trovano sotto la category Analog ed hanno i seguenti doctype: Design Specification, Design implementation, Signoff, Methodologies.
2. Il controllo passa al caso d'uso “recover documentation”, come parametri verranno passati category=”Analog”, doctype=”Design Specification”, “Design implementation”, “Signoff”, “Methodologies” e Family=“pll”, “clockgen”, “technology”.
3. I record recuperati vengono adesso ordinati per family e per doctype, e quindi visualizzati.

Use case name: “User Guide of tool used by Analog”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai tool contenuto nella pagina del gruppo funzionale Analog è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata..

Use case “included”: “Recover tool EDA user guide”.

Flow of events:

1. I tool sono gestiti dal gruppo funzionale EDA, quindi per quanto riguarda Analog esso potrà visualizzare solo i manuali d’uso dei tool da esso utilizzati.
2. Il controllo passa al caso d’uso “recover tool user guide ” al quale viene passato il nome del gruppo funzionale Analog.
3. I record trovati verranno raggruppati per famiglie (commercial EDA tools, internal EDA tools), e quindi visualizzati.

Use case name: “Project by Analog”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai progetti contenuto nella pagina del gruppo funzionale Analog è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata.

Use case “included”: “recover documentation”.

Flow of events:

1. La gestione della documentazione riguardante i Project prodotti dal team Analog avviene tramite “DIV-Web”.
2. Per recuperare tutti i documenti desiderati si deve fare una ricerca nel database, per recuperare tutti quei documenti che si trovano sotto la category Analog ed hanno un valore ben definito nel campo Project file;
3. Il controllo passa al caso d’uso “recover documentation”, come parametri verranno passati category=Analog, doctype= “all”, Project file diverso da NULL.
4. I record recuperati vengono adesso ordinati per Project file, doctype e per famiglie, e quindi visualizzati.

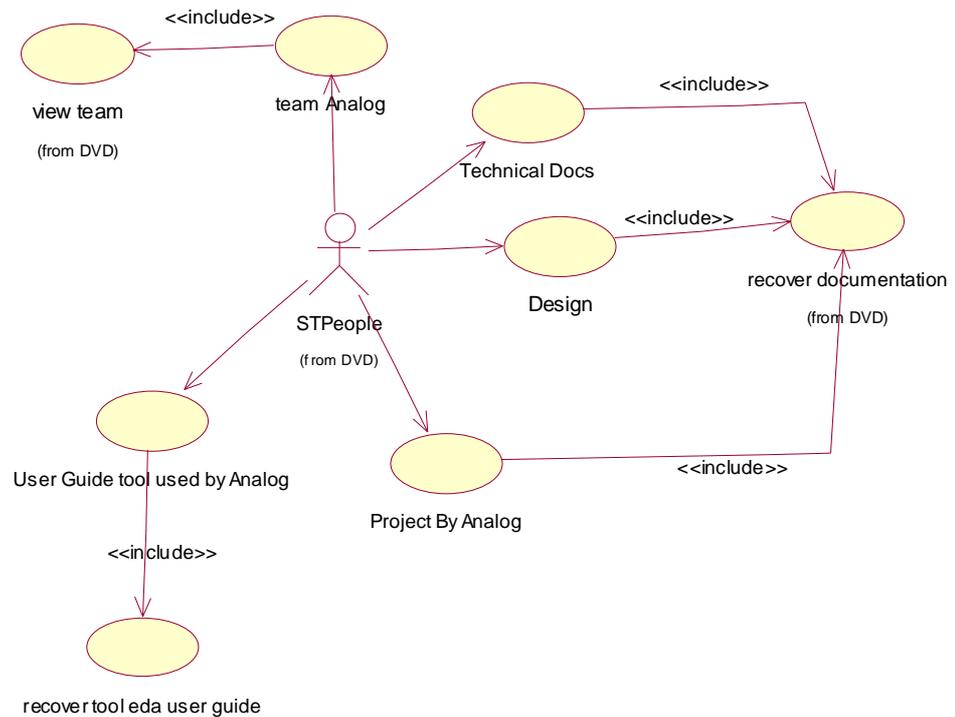


Figura 6.5 Diagramma dei Casi d'uso (Package Analog)

I casi d'uso del package Training, adesso, descritti e mostrati in figura 6.6.

<p>Use case name: “Team TRAINING”.</p> <p>Actors: STPeople.</p> <p>Priority: 1.</p> <p>Status: Requirements capture.</p> <p>Pre-condition: Il link “Team Training” contenuto nella pagina del gruppo funzionale Training è stato selezionato.</p> <p>Post-condition: L’elenco dei membri appartenenti al team è visualizzato.</p> <p>Use case “included”: “View Team”.</p> <p>Flow of events:</p> <ol style="list-style-type: none"> 1. Si recupera il nome del gruppo che contiene il team TRAINING. 2. Il controllo passa al caso d'uso “view team”, al quale viene passato il nome del gruppo in questione.
--

Use case name: “Training and information”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai Corsi contenuto nella pagina del gruppo funzionale Training è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata..

Use case “included”: “recover documentation”.

Flow of events:

1. La gestione della documentazione riguardante i corsi avviene tramite “DIV-Web”.
2. Per recuperare tutti i documenti riguardanti i corsi si deve fare una ricerca nel database, per recuperare tutti quei documenti che si trovano sotto la category Training ed hanno i seguenti doctype: Course, Tecnical Talks, Tools, Internal Design Rules, Docs and Manual;
3. Il controllo passa al caso d’uso “recover documentation”, come parametri verranno passati category=Training, doctype= Course, Tecnical Talks, Tools, Internal Design Rules, Docs and Manual.
4. I record recuperati vengono adesso ordinati per family(Front_End, Back_end, Verification, DFT/DFM, Design Flow, Design Theory, Design Reuse, Quality, Project Control, System, Unix, Word Processor), doctype, e quindi visualizzati. In questo caso le family rappresentano le aree d’interesse.

Use case name: “Training Forms”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition:Il link relativo Training Forms ai contenuto nella pagina del gruppo funzionale Training è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata..

Use case “included”: “recover documentation”.

Flow of events:

1. Le Training Forms sono gestite tramite “DIV-Web”.
2. Per visualizzarle si deve fare una ricerca nel database, per recuperare tutti quei documenti che si trovano sotto la category Training e doctype Training forms.
3. Il controllo passa al caso d’uso “recover documentation”, come parametri verranno passati category=Training, doctype=TrainingForms.
4. I record recuperati vengono adesso ordinati per family (people, template), e quindi

visualizzati.

Use case name: “Course Providers”.

Actors: STPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: Il link relativo ai Course Provider contenuto nella pagina del gruppo funzionale Training è stato selezionato.

Post-condition: La documentazione è adesso disponibile per essere visualizzata..

Use case “included”: “Recover documentation”.

Flow of events:

1. La gestione della documentazione riguardante i fornitori dei corsi avviene tramite “DIV-Web”.
2. Si deve fare una ricerca nel database, per recuperare tutti quei documenti che si trovano sotto la category Training e doctype Course Provider.
3. Il controllo passa al caso d’uso “recover documentation”, come parametri verranno passati category=Training, doctype=Course Provider.
4. I record recuperati vengono adesso ordinati per family (Internal, STU, External), e quindi visualizzati.

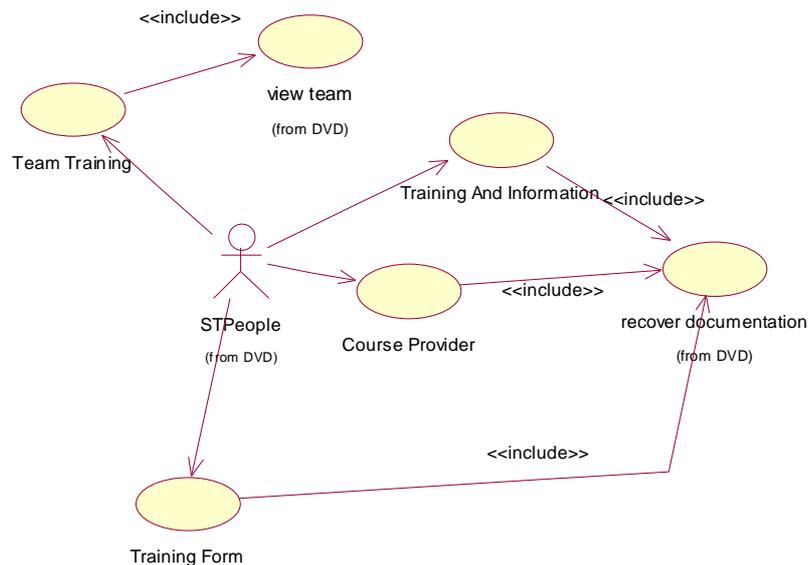


Figura 6.6 Diagramma dei Casi d’uso (Package Training)

L'analisi dei casi d'uso finisce con quelli appartenenti al package Notice Board, visibili in figura 6.7.

Use case name: "Modify NOTICE".

Actors: Owner.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "Modify" è stata selezionata da "Update Notice".

Post-condition: L'evento selezionato è stato modificato.

Flow of events:

1. Un form, che consentirà la modifica dell' evento della "notice board" selezionato, è adesso visualizzato, nel campo di testo dedicato all'inserimento dei nuovi dati sono presenti quelli già memorizzati, pronti per essere modificati.
2. L'owner modifica ciò che è contenuto nei vari campi.
3. Clicca sul bottone "Modify".
4. Evento modificato.

Secondary scenario:

3. Clicca sul bottone "Cancel".
4. Si ritorna alla pagina precedente.

Use case name: " Insert New Notice".

Actors: CMGpeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'utente è già stato autenticato. Solo un dipendente CMG può inserire un nuovo evento in bacheca.

Post-condition: Un nuovo evento è stato inserito.

Flow of events:

1. Il form per l'inserzione di un nuovo evento è adesso visibile.
2. Tramite il form si potranno inserire le seguenti informazioni:
 - a) Titolo dell' evento;
 - b) Un eventuale file allegato;
 - c) La data di scadenza;
 - d) Data dell' evento;
 - e) Visibilità (CMG o STworld);
 - f) Luogo dell'evento;
 - g) Il contenuto dell'annuncio;
3. Nel form è presente anche un bottone tramite il quale si può confermare l'inserzione.

4. Se l'inserzione viene confermata l'annuncio sarà memorizzato.

Secondary scenario:

3. Inserzione non confermata.

4. Annuncio non memorizzato.

Use case name: "Update notice".

Actors: Owner.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'utente è già stato autenticato. Esso potrà modificare esclusivamente gli eventi di sua proprietà.

Use case "included": "Modify Notice", "Delete Notice".

Flow of events:

1. Tutti gli eventi inseriti dell'owner sono adesso visualizzati.
2. Su ogni annuncio esistente si possono effettuare due operazioni, a scelta: modifica e cancellazione.
3. Se viene scelta l'azione:
 - a. "modifica", in corrispondenza ad un evento già esistente, il controllo passa al caso d'uso "Modify Notice", al quale viene passato l'evento da modificare.
 - b. "delete", in corrispondenza ad un evento già esistente, il controllo passa al caso d'uso "Delete Notice", al quale viene passato l'evento da cancellare.

Use case name: "Delete NOTICE".

Actors: Owner.

Priority: 1.

Status: Requirements capture.

Pre-condition: L'azione "Delete" è stata selezionata da "Update Notice".

Post-condition: L'evento della notice board selezionato è stato cancellato.

Flow of events:

1. Una finestra di dialogo, che chiede conferma per la futura cancellazione dell'evento della notice board selezionato, è adesso visualizzata.
2. L'owner clicca sul bottone "yes".
3. L'evento è stato cancellato.

Secondary scenario:

2. Clicca sul bottone "no".

3. Si ritorna alla pagina precedente.

Use case name: “Authentication”.

Actors: STPeople, CMGPeople, Owner.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’utente vuole accedere alla Notice Board.

Post-condition: L’utente è stato riconosciuto dal sistema.

Flow of events:

1. Il caso d’uso inizia quando è visualizzata la finestra che consente l’inserimento di login e password.
2. In questa finestra ci sono due bottoni: “ok” e “cancel”.
3. Se l’utente seleziona il bottone “ok”:
 - a) Il sistema si collega al server ldap;
 - b) Si ricerca un record che contenga la Login fornita dall’utente, nella “Enterprise Directory”.
 - c) Si verifica se la Password inserita coincide con quella memorizzata nel record appena prelevato.
 - d) Si controlla se nel campo relativo alla organizzazione è presente CMG.
 - e) Se è presente allora l’attore sarà riconosciuto come CMGPeople, altrimenti sarà riconosciuto come STPeople.
 - f) Se l’attore è un CMGPeople ma risulta proprietario di qualche evento esso sarà riconosciuto come Owner.

Secondary scenario:

- A. 3. b) Il server ldap non è disponibile.
 - c) L’utente non si potrà autenticare.
- B. 3. b) La login fornita è sbagliata.
 - c) L’utente non sarà autenticato deve riprovare di nuovo ad inserire una login giusta.
- C. 3. b) La Password fornita è sbagliata.
 - c) L’utente non sarà autenticato deve riprovare di nuovo ad autenticarsi.

Use case name: “view Notice”.

Actors: CMGPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’utente è autenticato, appartiene al gruppo CMG Catania.

Use case included: “View Notice CMG”, “View Notice ST”.

Flow of events:

1. Tutti gli eventi memorizzati sono visualizzati.
2. Suddivisi in due categorie :
 - a) eventi interni visualizzati tramite il caso d’uso “view notice CMG”.
 - b) eventi esterni visualizzati tramite il caso d’uso “view notice ST”.

Use case name: “view Notice CMG”.

Actors: CMGPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’utente è autenticato, appartiene al gruppo CMG Catania.

Post-condition: L’elenco degli eventi CMG è adesso disponibile.

Flow of events:.

1. Tutti gli eventi marcati come non visibili sono visualizzati.
2. Per ogni evento sarà visibile il Titolo, il file allegato, l’owner, la data dell’evento, e il testo.
3. Gli eventi sono ordinati per data evento in modo crescente.

Use case name: “view Notice ST”

Actors: STPeople, CMGPeople.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’utente della notice board è autenticato.

Post-condition: L’elenco degli eventi visibili è adesso disponibile.

Flow of events:

1. Tutti gli eventi marcati come visibili sono visualizzati.
2. Per ogni evento sarà visibile il titolo, il file allegato, l’owner, la data dell’evento, e il testo.
3. Gli eventi sono ordinati per data evento in modo crescente.

Use case name: “Notify expiration”.

Actors: Cronjob.

Priority: 1.

Status: Requirements capture.

Pre-condition: La data e l’ora memorizzate nel cronjob sono scattate.

Post-condition:.

Flow of events:.

1. La data corrente viene confrontata con la data evento di tutti gli eventi memorizzati.
2. Se queste due date coincidono verrà mandata un e-mail all’owner per notificare la prossima scadenza dell’ evento e dare la possibilità di rinnovarlo.

Use case name: “Confirm/refuse expiration”.

Actors: Owner.

Priority: 1.

Status: Requirements capture.

Pre-condition: L’e-mail inviata dal cronjob tramite “Notify expiration notice to owner” è arrivata all’owner.

Flow of events:

1. L'owner, tramite l'e-mail inviata dal sistema per notificare la prossima scadenza di un evento, può rinnovare l'evento, oppure può non fare nulla.

Use case name: "Clean up".

Actors: Cronjob.

Priority: 1.

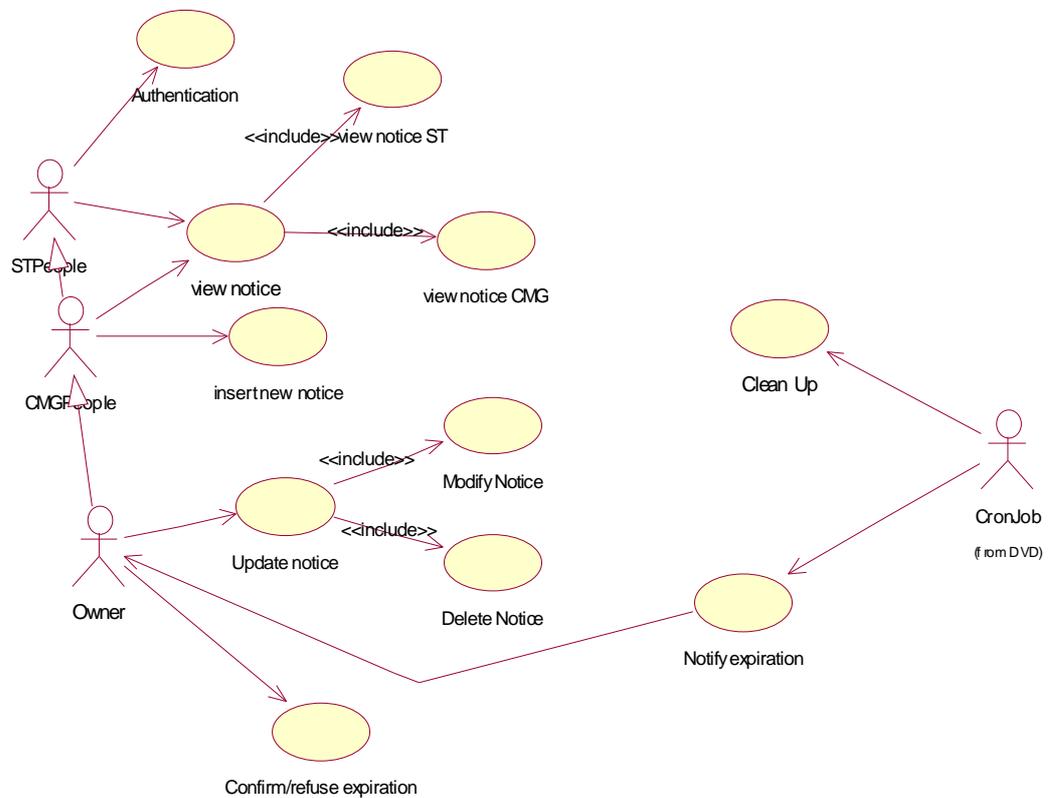
Status: Requirements capture.

Pre-condition: La data e l'ora memorizzate nel cronjob sono scattate.

Post-condition: Gli eventi scaduti sono stati cancellati.

Flow of events:.

1. La data corrente viene confrontata con la data di scadenza di tutti gli eventi memorizzati.
2. Raggiunta la data di scadenza di un evento esso sarà cancellato automaticamente.



Le Figura 6.7 Diagramma dei Casi d'uso (Package Notice Board)

5 Analisi e Design dell'applicazione “DVD-STB Web”

In questo paragrafo verranno descritti i diagrammi tramite i quali saranno analizzate le caratteristiche dell'applicazione sotto vari punti di vista. Saranno mostrati i diagrammi: concettuale, di navigazione, e di presentazione. Per la costruzione di questi diagrammi è stata adottata la metodologia introdotta nel Capitolo 1.

5.1 Diagramma Concettuale

Il diagramma concettuale è il diagramma del dominio del problema, tramite esso vengono rappresentati tutti i concetti che sono rilevanti per l'applicazione in esame. Il principale obiettivo è quello di catturare la semantica del dominio. Gli aspetti riguardanti la navigazione e la presentazione verranno trattati nei paragrafi relativi.

La suddivisione in package vista in figura 6.1, rimarrà invariata nei paragrafi seguenti. Questo tipo di suddivisione rende più facile la comprensione e la visione dei modelli.

Iniziamo l'analisi del modello dal package DVD.

5.1.1 Package DVD

In figura 6.8 si può notare che il diagramma concettuale per il package DVD. La classe principale è “DVD_STB_Web” modella le divisioni DVD e STB.

Ha una relazione di composizione con la classe “CONFIGURATION”, perché quest'ultima modella le configurazioni necessarie per lo sviluppo dell'applicazione in studio. Questa classe ha i seguenti attributi:

- *Name-division* indica il nome della divisione che si sta rappresentando.
- *Location* “site ST” della divisione.
- *Standards* standard aziendali per rappresentare le pagine Web.

Una divisione è composta da gruppi funzionali, quindi la classe “DVD_STB_Web” ha una relazione di composizione con la classe “FUNCTIONAL_GROUP” la quale modella i gruppi funzionali che compongono la divisione DVD_STB.

In questa tesi, verranno trattati i gruppi funzionali EDA, Timing Analysis, Analog, Training. I rimanenti gruppi saranno sviluppati in futuro.

Le classi che modellano EDA, Timing Analysis, Analog, Training sono delle generalizzazioni di FUNCTIONAL_GROUP. Queste specializzazioni sono state omesse in figura 6.8 ma saranno visibili nelle altre figure.

I FUNCTIONAL_GROUP sono composti da team (gruppi di persone modellati dalla classe “team-group”).

I “team-group” hanno le seguenti relazioni:

1. **Dipendenza** con la classe “GROUP” che modella i gruppi che vengono formati tramite l'applicazione “The Group Service”.
2. **Composizione** con la classe “People”, la quale conterà tutti i dati rilevanti per la nostra applicazione che riguardano le persone appartenenti ai vari team.
3. **Specializzazioni**: “team_project”, “team_eda”, “team_ta”, “team_training”, “team-analog”.

La classe “People” ha una relazione di dipendenza con la classe “Enterprise Directory”. Questo sta ad indicare che tutti i dati delle “People” vengono recuperati interrogando l’applicazione Enterprise Directory. Per ogni “People” verranno recuperate le seguenti informazioni :

- Username.
- Password.
- Name (nome e cognome).
- Mail.
- Phone.
- Location (sito ST di appartenenza).

La classe “team leader” è una specializzazione di “People” che modella il responsabile di un certo gruppo di persone.

La classe “User” individua l’utente che accederà a “DVD-STB Web”. Esso può essere un “STPeople” senza nessuna identità, oppure una delle “People” (“TeamLeader”) appartenenti ai vari team le quali avranno dei permessi particolari. Una specializzazione di “People” è “TeamLeader” che rappresenta i responsabili dei “team” modellati tramite “team_group”.

La classe “Project_info” modella tutte le informazioni rilevanti per un progetto. Essa ha due relazioni di composizione:

1. *Team-project*. Specializzazione di team-group, ha la stessa struttura vista prima.
2. *Project Documentation*. Dipendente da DVD, rappresenta tutta la documentazione legata ad un progetto gestita dall’applicazione “Div-Web”.

La classe “DVD” rappresenta la tabella del database di “Div-Web” per le divisioni DVD e STB. Questa classe non verrà implementata, essa già esiste ed è gestita in un altro *site*ST.

Gli attributi della classe DVD si possono vedere in figura 6.8, tra i quali quelli più rilevanti per l’applicazione in fase di studio sono:

- *Author*: rappresenta l’autore del documento.
- *Date* data di inserimento.

- *Category*: indica la categoria del documento.
- *DocType*: tipo di documento.
- *Adcs*: collegamento ad un documento ADCS.
- *Family*: famiglia di documenti di appartenenza.
- *Description*: descrizione associata.
- *Url*: url associata.
- *Filename*: nome del file associato.
- *Owner* e *Group owner*: responsabili del documento.
- *Project_file*: identificativo unico per progetto o tool.

“Project_documentation” ha una relazione di dipendenza con “DVD” in quanto le sue istanze sono ottenute tramite una ricerca fatta sulla tabella “DVD”, prelevando esattamente tutti i documenti che hanno come “Project_file” quello indicato nella classe “Project_Info”.

La classe “Operation_meeting” sarà ottenuta allo stesso modo di Project_documentation. Ma a differenza di “Project_documentation” essa andrà a ricercare tutti i documenti che avranno come category=”Management” e doctype=”Staff meeting Minutes”.

External_node indica tutte le classi esterne che saranno contattate dalla applicazione. Esse sono riportate in figura 6.9.

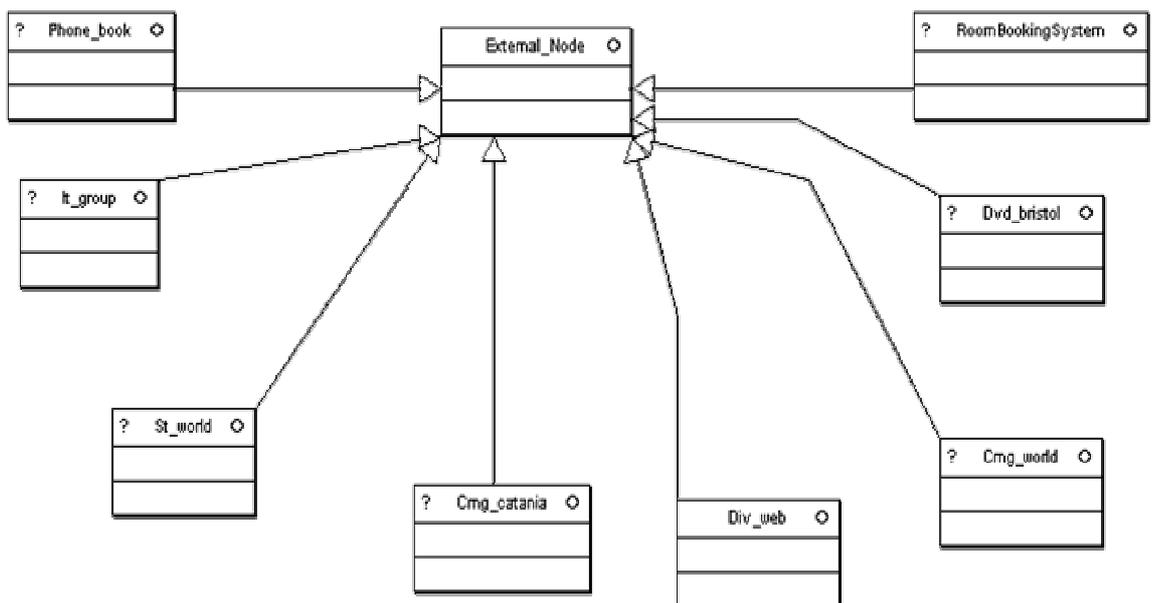
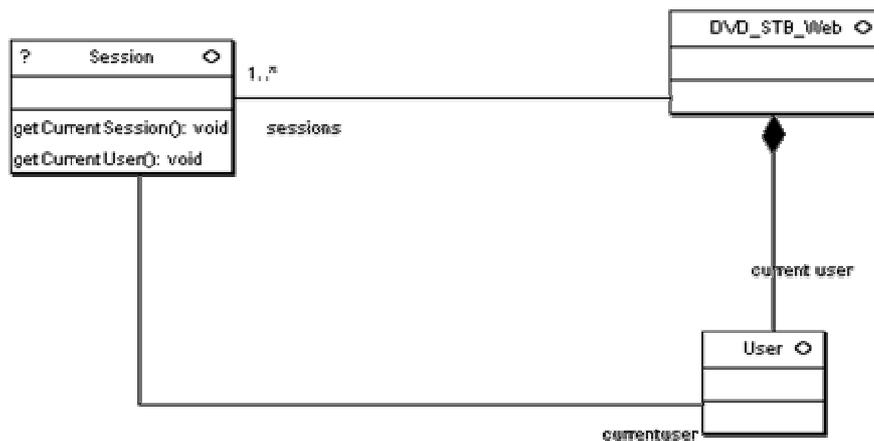


Figura 6.9. Pagine Web esterne che verranno collegate a DVD STB Web

“Notice_Board” appartiene al package omonimo, parleremo di essa nella sezione relativa.

“DIV_Web” e “The Group service” modellano le applicazioni omonime, esse sono delle classi già esistenti.



La classe “Session” tiene traccia dei dati necessari per un’intera esecuzione dell’applicazione. In particolare, individuerà l’identità dell’utente autenticato, teamleader, operazione necessaria affinché il teamleader possa entrare nell’area amministrativa dell’applicazione, come si vedrà nel vincolo OCL presente nei diagramma di navigazione in figura 6.16.

5.1.2 Package EDA

In figura 6.10 si può notare la specializzazione fra la classe “EDA” e “FUNCTIONAL-GROUP”. “EDA” ha quindi tutte le caratteristiche di “FUNCTIONAL-GROUP”. Per quanto riguarda il “team” l’approccio seguito è lo stesso visto nel package DVD, in quanto “Team-EDA” è una specializzazione di “Team-group”.

“License expiration” è una classe che modella la gestione delle scadenze delle licenze dei tool commerciali.

Le classi “internal-tool-documentation”, “commercial-tool-documentation”, “design-documentation”, sono dipendenti da “DVD”, ed in particolare:

- “internal-tool-documentation” è formata da documenti di “DVD” che appartengono alla category=”EDA”, docType=”Tool development Plan and report” o “Tool release notes” o “Tool Specification” o “Evaluation spec/report” o Procedures”, family=”internal EDA Tools”.
- “commercial-tool-documentation” è formata da documenti di “DVD” che appartengono alla category=”EDA”, docType=”Tool development Plan and report” o “Tool release notes” o “Tool Specification” o “Evaluation spec/report” o Procedures”, family=”commercial EDA Tools”.
- “Design documentation” è formata da documenti di “DVD” che appartengono alla category=”EDA”, docType=”methodologies” o “design environment” .

5.1.3 Package Analog

Le classi “FUNCTIONAL GROUP”, “team-analog”, sono modellate come le classi relative al package “EDA”. Il diagramma relativo è mostrato in figura 6.11.

Le seguenti classi hanno una relazione di dipendenza con DVD:

- “Technical-talks”: è formata da documenti di “DVD” che appartengono alla category=”Analog”, docType=”Technical-talks”.
- “Analog_design”: è formata da documenti di “DVD” che appartengono alla category=”Analog”, docType=”designspecification”, “design_implementation”, “methodologies”.
- “Sign_off”: è formata da documenti di “DVD” che appartengono alla category=”Analog”, docType=”signoff”.
- “Project”: è formata da documenti di “DVD” che appartengono alla category=”Analog”, che hanno associato un identificativo di progetto.
- “Tool_user guide”: verrà utilizzata da un punto di vista Analog per recuperare i manuali dei tool gestiti da “EDA” ma utilizzati da Analog. Le istanze della classe “tool_user guide” saranno tutti i documenti che appartengono alla category=”EDA”, doctype=”Tool user guide” , user=”Analog” o “ALL”.

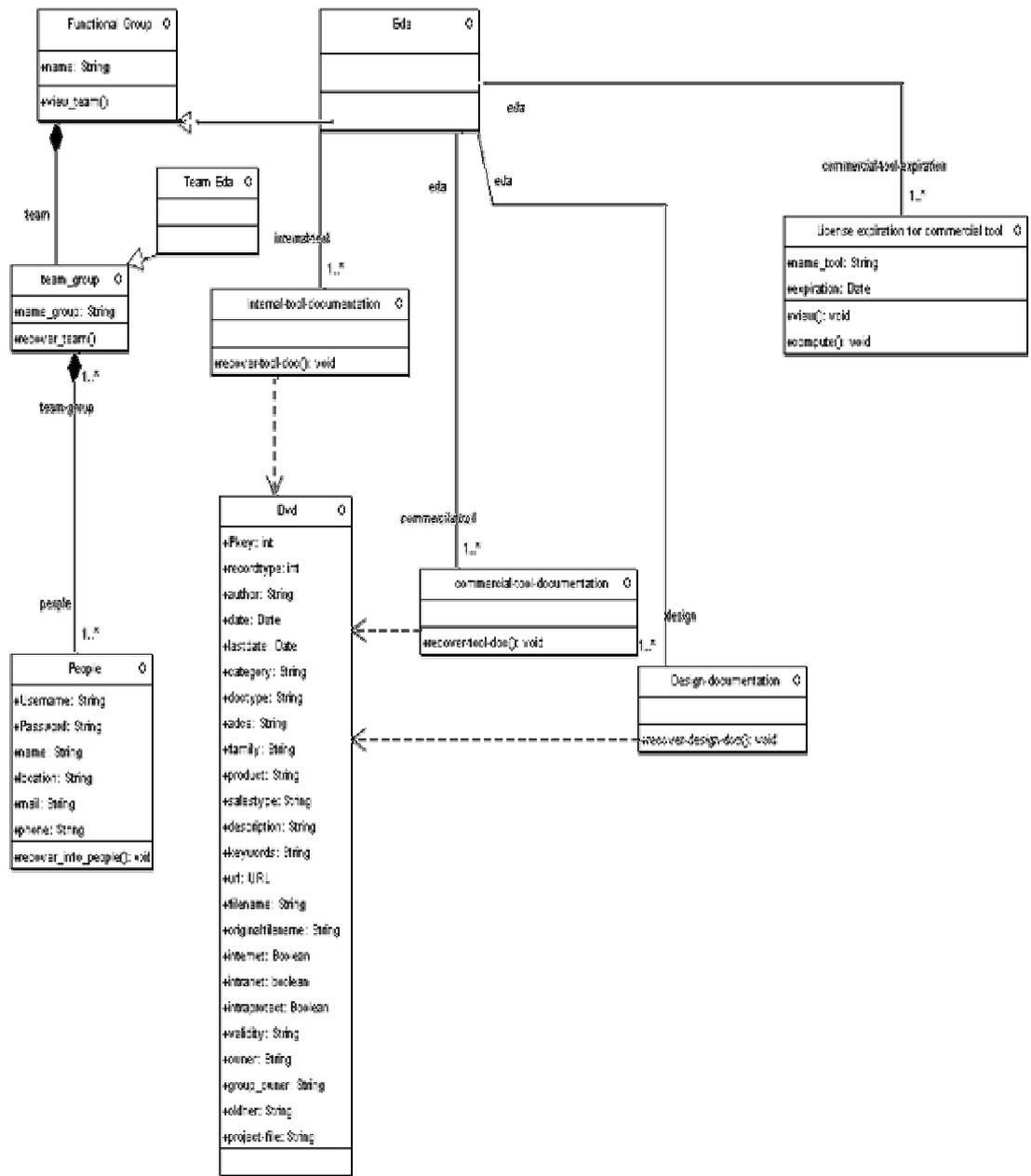


Figura 6.10 Diagramma concettuale Eda

5.1.4 *Package Timing Analsis*

Le classi “FUNCTIONAL GROUP”, “team-Timing Analysis”, sono modellate come le classi relative al package “EDA”. Il diagramma relativo è mostrato in figura 6.12.

Le seguenti classi hanno una relazione di dipendenza con DVD:

- “Script”: è formata da documenti di “DVD” che appartengono alla category=”Timing Analysis”, docType=”script” .
- “Sign_off Criteria”: è formata da documenti di “DVD” che appartengono alla category=”Timing Analysis”, docType=”SignOff Criteria”.
- “Sign_off”: è formata da documenti di “DVD” che appartengono alla category=”Timing Analysis”, docType=”signoff”.
- “Tool_user guide”: verrà utilizzata da un punto di vista Timing Analysis per recuperare i manuali dei tool gestiti da “EDA” ma utilizzati da Timing Analysis. Le istanze della classe “tool_user guide” saranno tutti i documenti che appartengono alla category=”EDA”, doctype=”Tool user guide” , user=”Timing Analysis” o “ALL”.

5.1.5 *Package Training*

Per le classi “FUNCTIONAL GROUP”, “team-Training”, sono modellate come le classi relative al package “EDA”. Il diagramma relativo è mostrato in figura 6.13.

Le seguenti classi hanno una relazione di dipendenza con “DVD”:

- “Catalog Course”: è formata da documenti di DVD che appartengono alla category=”Training”, docType=”Course” o “Technical Talks” o “Tools” o “Internal Design Rules” o “Docs and Manual”;
- “Course Provider”: è formata da documenti di “DVD” che appartengono alla category=”Training”, docType=”Course Provider”.
- “Training Forms” è formata da documenti di DVD che appartengono alla category=”Training”, docType=”Training Forms”.

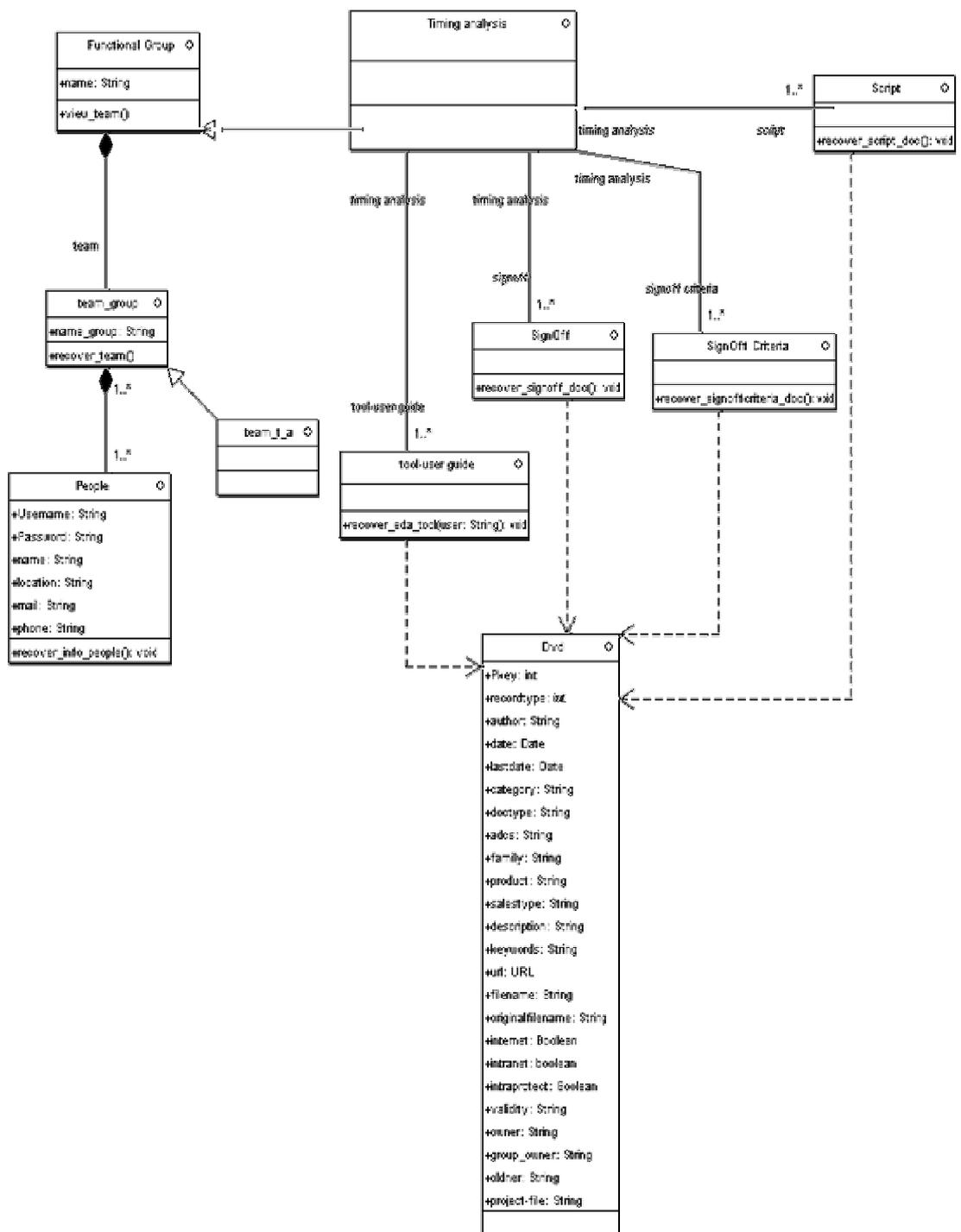


Figura 6.12. Diagramma concettuale TimingAnalysis

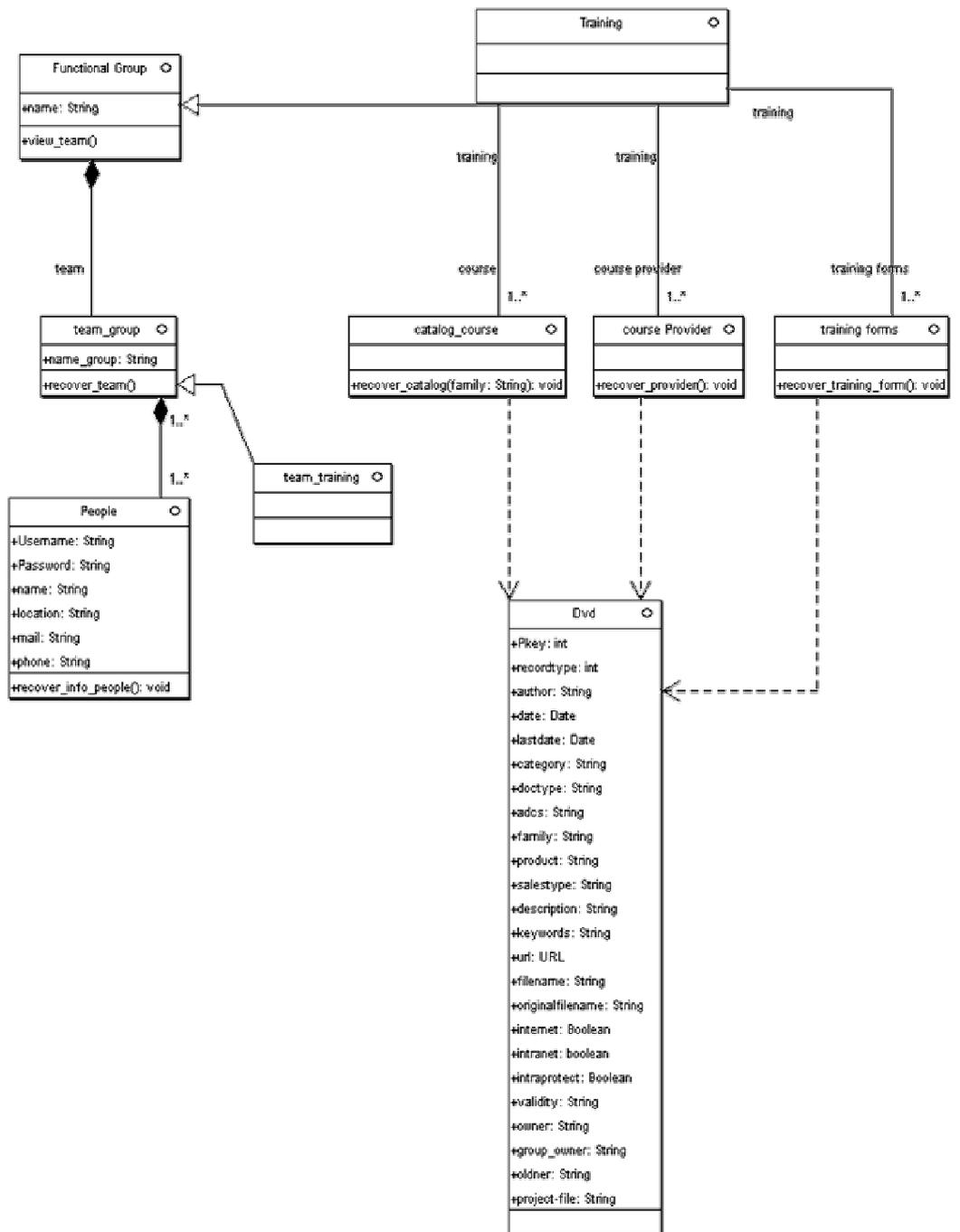


Figura 6.13. Diagramma concettuale Training

5.1.6 *Package Notice Board*

La classe “NOTICE-BOARD” modella la bacheca elettronica. Essa ha delle associazioni di composizione con la classe “Notice” che rappresenta l’evento vero e proprio. Gli attributi di “Notice” sono:

- *Title* Indica il titolo dell’evento.
- *Day_of_creation* Giorno dell’inserimento.
- *Day_of_expiration* Giorno di scadenza.
- *Day_event* Giorno dell’evento.
- *Visible* Indica se l’evento può essere visibile da tutta la STMicroelectronics o solo dalla divisione CMG Catania.
- *Location* Luogo dell’evento.
- *Body* Descrizione dell’avvenimento.
- *Attach* File allegato.
- *URL* Url associata.
- *Day_update* Giorno in cui l’evento è stato aggiornato.

La classe “User” modella i tre tipi d’utenti della “Notice Board”:

1. “STPeople”: il quale può esclusivamente leggere gli eventi contrassegnati come *visibili*.
2. “CMGPeople”: potrà leggere tutti gli eventi (visibili e non) inseriti nella “Notice Board”, ed eventualmente potrà inserirne uno nuovo.
3. “Owner”: è un “CMGPeople” che ha inserito qualche evento, egli potrà modificare o cancellare un evento da lui inserito.

La classe “Session” tiene traccia dei dati necessari per un’intera esecuzione dell’applicazione. In particolare, individuerà l’identità dell’utente della sessione corrente come si vedrà nei vincoli OCL presenti nei diagrammi di navigazione (Figure 6.27,6.28, 6.29).

La classe “Notify” modella la notifica di scadenza dell’evento che verrà inviata automaticamente al proprietario. (Quest’azione viene effettuata tramite un cronjob, quindi non sarà modellata nei diagrammi di navigazione).

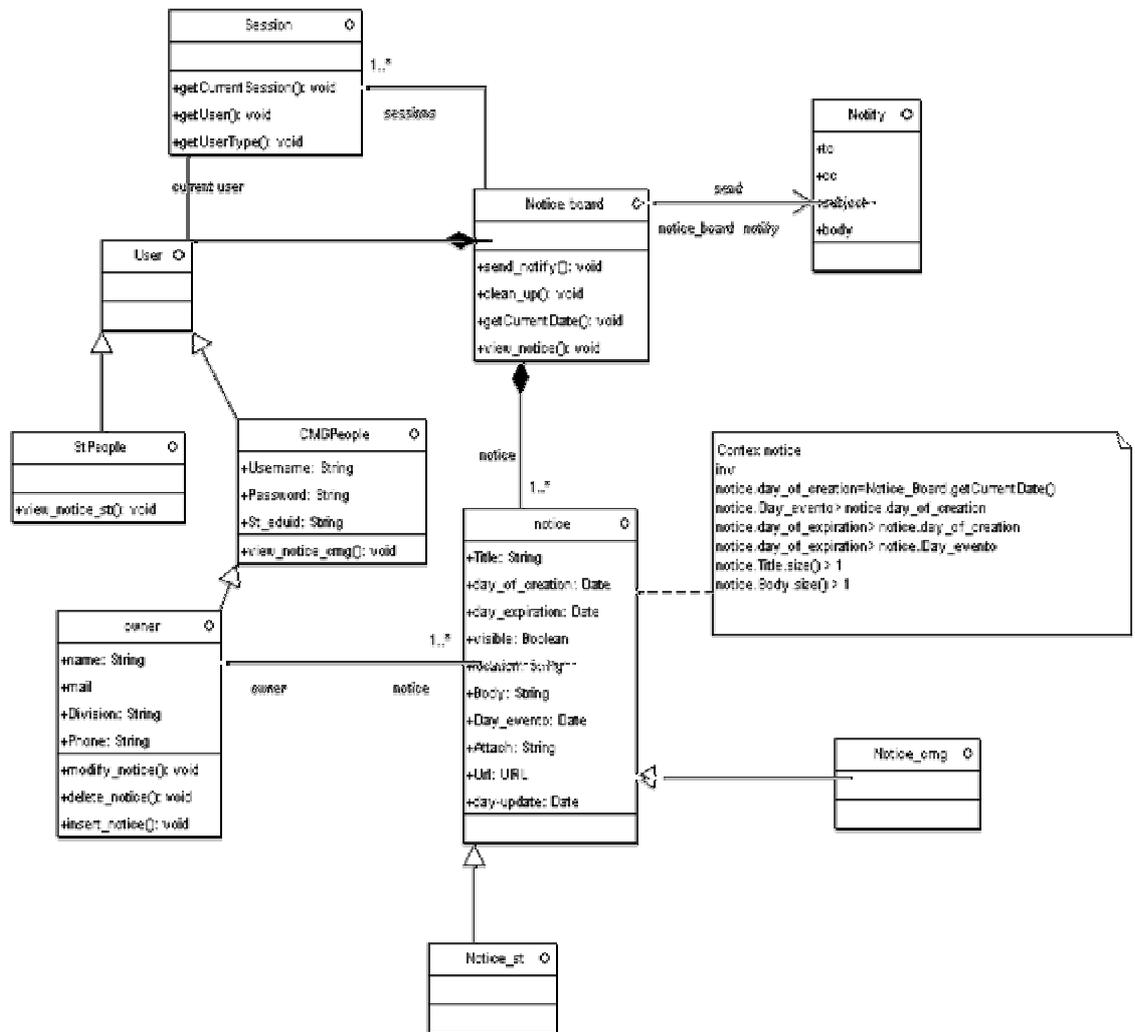


Figura 6.14 Diagramma concettuale (Package Notice Board)

5.2 Diagramma di navigazione

Il modello di navigazione dell'applicazione in fase di studio è composto:

1. Dal modello dello spazio di navigazione e che mostra *quali* oggetti possono essere raggiunti mediante una navigazione.
2. Dal modello della struttura di navigazione e mostra *come* saranno raggiunte le classi di navigazione.

Il punto di partenza per la costruzione di questo modello è il modello concettuale. Verranno considerate solo quelle classi concettuali che saranno importanti per la navigazione, cioè quelle che diventeranno le nostre pagine Web. Alle associazioni presenti nel modello concettuale, ne vengono aggiunte altre che saranno necessarie per realizzazione dei casi d'uso.

In figura 6.15 possiamo vedere il diagramma dello spazio di navigazione per il *Package DVD*.

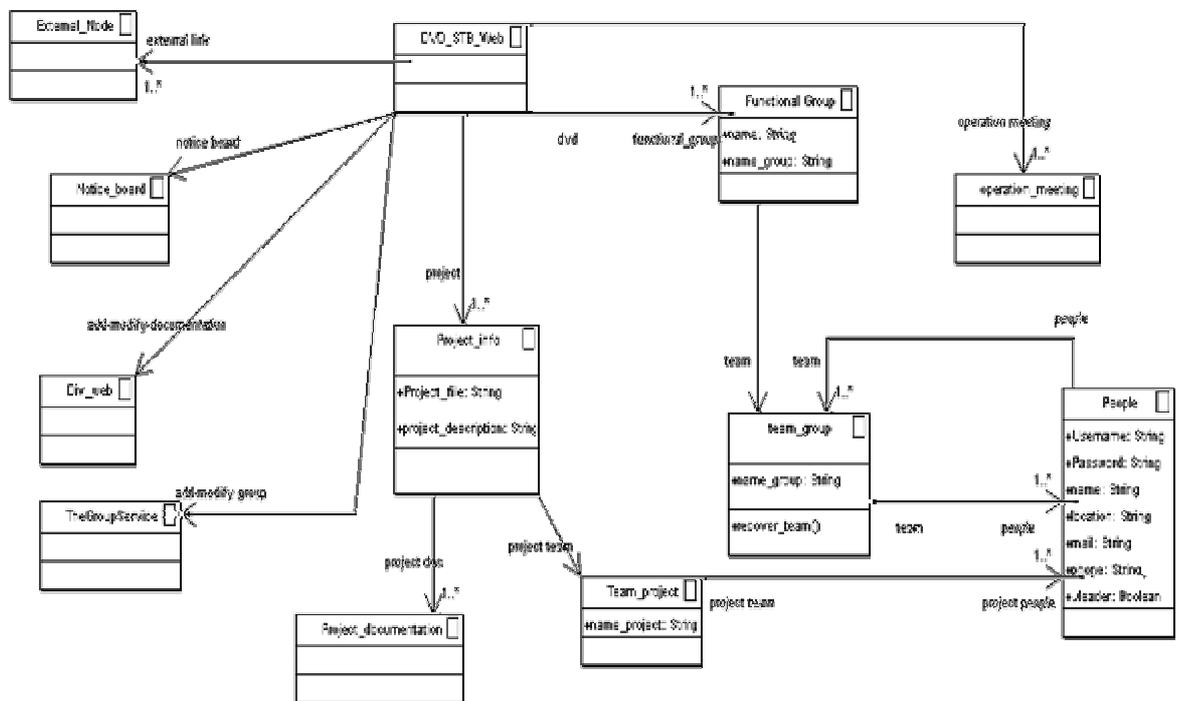


Figura 6.15 Diagramma dello spazio di navigazione Package DVD

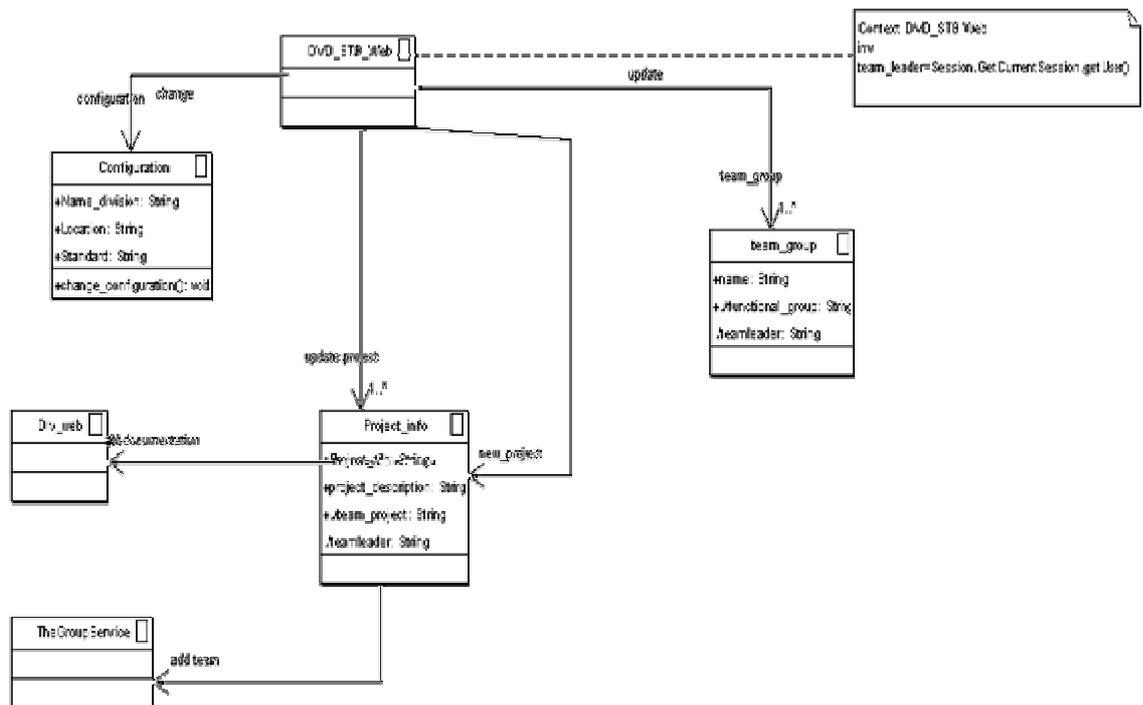


Figura 6.16 Diagramma dello spazio di navigazione Package DVD(vista teamleader)

Le classi non rilevanti per la navigazione sono state omesse, cioè: “Group”, “User”, “STPeople”, “Enterprise Directory”, “DVD”, “DIV-Web-DB” “Table”.

[Nota: La classi “Div_web”, “The Group Service”, “External Node” corrispondono ai “Nodi Esterni” definiti nel capitolo 1.]

Le classi dipendenti da “DVD” ereditano tutti i suoi attributi, per ragioni di spazio e di visibilità questi attributi non saranno visibili nelle figure.

Per soddisfare tutti i casi d'uso, nella vista DVD, è stata introdotta l'associazione fra: “Project-info” e “team-project”, per consentire la visione del team relativo al progetto.

Analizzando, quindi, il diagramma in figura 6.15 si può notare che dall'*home page* “DVD_STB Web” si può:

1. Raggiungere una delle pagine appartenenti agli “External node”.
2. Accedere al “Notice Board”.
3. Aggiungere o modificare un documento accedendo all'applicazione Div-Web.
4. Aggiungere o modificare un gruppo di persone accedendo all'applicazione The Group Service.
5. Vedere le informazioni che riguardano i vari progetti: documentazione e team relativo.
6. Accedere alle pagine dei gruppi funzionali.
7. Leggere gli “operation meeting”.

In figura 6.16 si può vedere lo spazio di navigazione relativo all'area amministrativa a cui potrà accedere solo il “Team Leader”. Entrando nell'area amministrativa il team leader potrà:

1. Inserire un nuovo progetto.
2. Aggiornare un progetto esistente.
3. Cambiare le configurazioni.
4. Modificare il nome dei vari “team_group” ed il relativo “teamleader”.

In particolare, si possono notare le associazioni introdotte fra:

1. “Project_info” e “DiV-Web” per soddisfare il caso d'uso “New project”.
2. “Project-info” e “The Group Service” per soddisfare il caso d'uso “New project”.

I diagrammi in figura 6.15 e 6.16, come già detto, mostrano quali classi verranno raggiunte dalla navigazione. Questi sono stati arricchiti mediante l'aggiunta delle primitive d'accesso per mostrare come avverrà la navigazione, ciò è mostrato in figura 6.17 e 6.18 dal diagramma della struttura di navigazione.

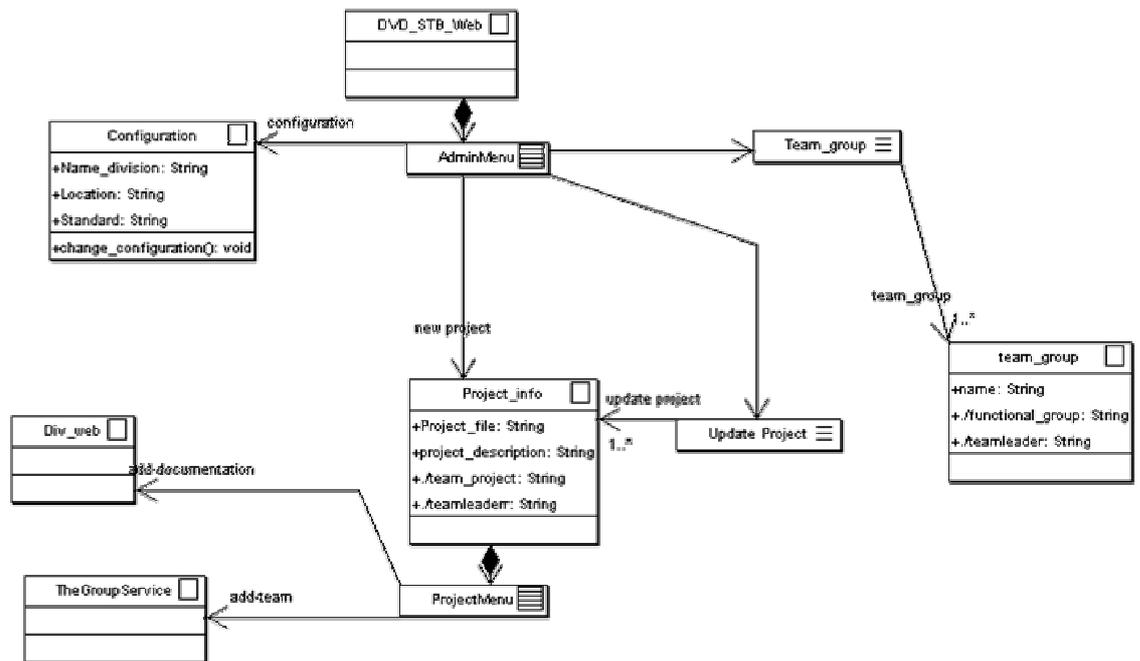


Figura 6.18 Diagramma della struttura di navigazione Package DVD(vista relativa al teamleader)

Sono state aggiunte le seguenti primitive d'accesso: index, query e menu. In particolare, i due menu aggiunti, in figura 6.17, sono:

1. “DVD_STB Web Main Menu”, tramite il quale si potranno effettuare tutte le azioni descritte prima.
2. “Project Menu”, tramite il quale si potranno effettuare tutte le azioni legate ad un progetto.

Tramite gli “index“ in fase di navigazione, possiamo scegliere ed accedere alle varie istanze di una classe di navigazione.

L’index “Functional Group” consente all’utente di scegliere quale gruppo funzionale visitare.

Il query “Operational Meeting by year” permette di ordinare e quindi estrarre solo gli “operation meeting” di un determinato anno, tramite l’index associato si sceglierà quello desiderato.

In figura 6.18, si può notare il menu “AdminMenu” dal quale si può scegliere se cambiare configurazione, o aggiungere un nuovo progetto ed inserire le varie documentazioni e team, scegliere il progetto da modificare, o cambiare i nomi dei “team_group”. Il query “Project to Update” consentirà di estrarre le informazioni relative ad un progetto, da aggiornare.

Lo stesso processo effettuato per DVD, è stato attuato sui package EDA, TIMING ANALYSIS, ANALOG, TRAINING che possiamo vedere nelle figure 6.19, 6.21, 6.23, 6.25.

Per il package EDA, si può notare in figura 6.19, che dall’home page del gruppo funzionale EDA possono essere effettuate le seguenti azioni:

1. Visualizzare il team EDA.
2. Consultare la documentazione che riguarda il tool interni.
3. Consultare la documentazione che riguarda il tool commerciali.
4. Consultare la documentazione relativa i design prodotti da EDA.
5. Accedere alle date di scadenza delle licenze dei vari tool commerciali.

In figura 6.20, si può osservare il diagramma della struttura di navigazione relativo a quello in figura 6.19 nel quale sono stati introdotti il menu principale “Functional Group EDA Main Menu” ed i vari index per accedere alle varie istanze di “internal-tool-documentation”, “commercial-tool-documentation”, “design-documentation”, “License expiration for commercial tool”, “People”.

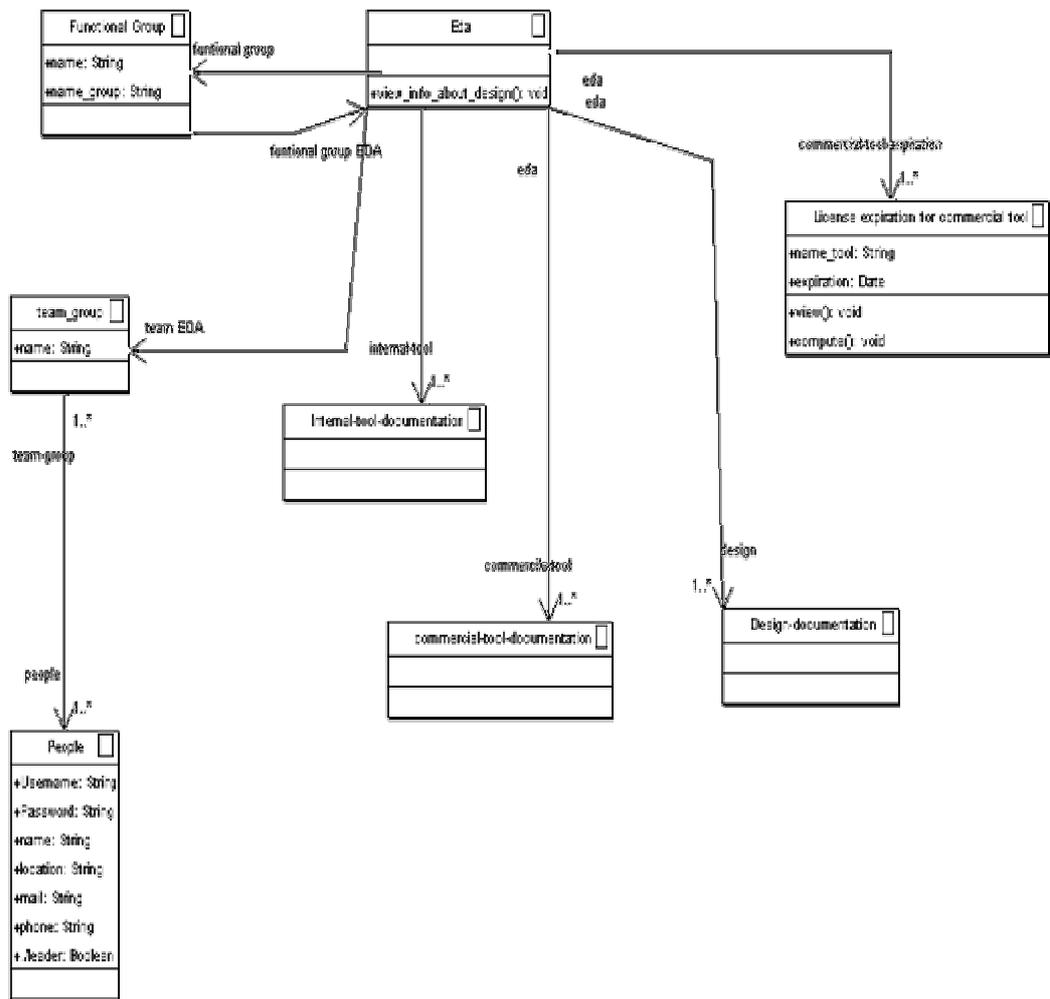


Figura 6.19 Diagramma dello spazio di navigazione Package EDA

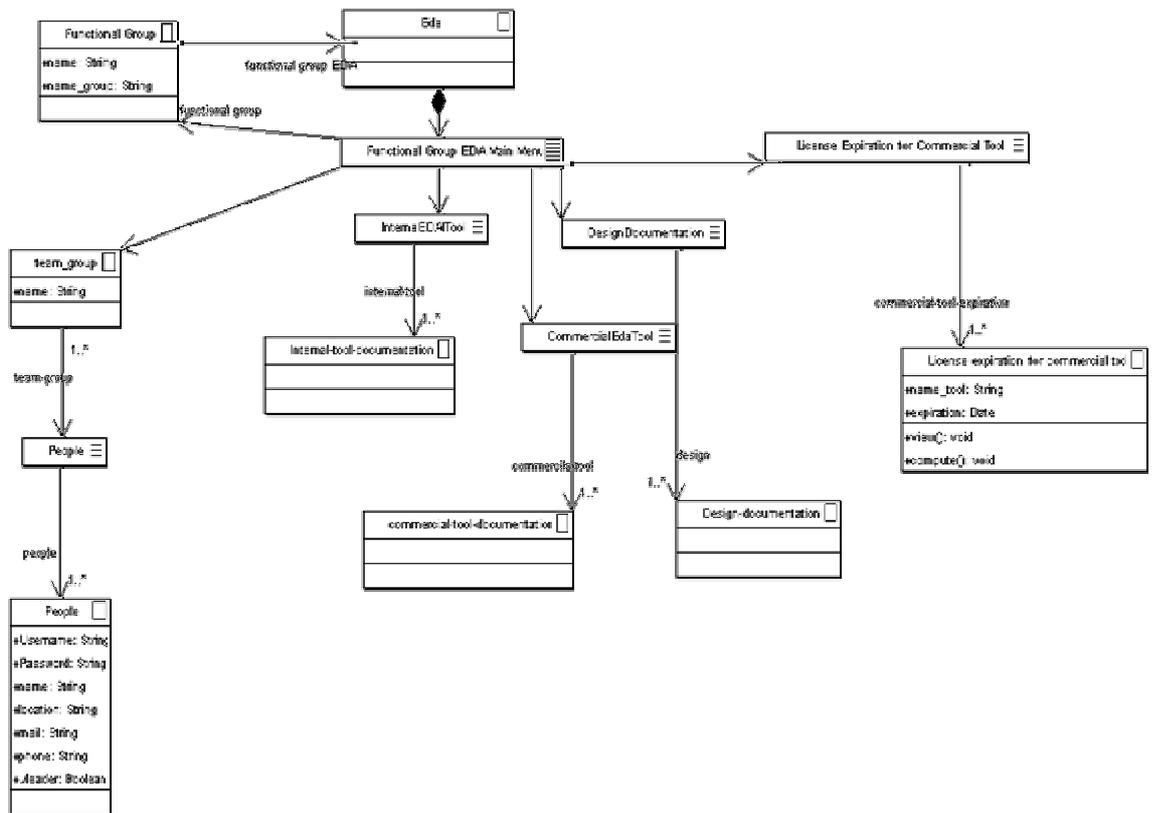


Figura 6.20 Diagramma della struttura di navigazione Package EDA

Per il package Timing Analysis, si può notare in figura 6.21, che dall’home page del gruppo funzionale Timing Analysis possono essere effettuate le seguenti azioni:

1. Visualizzare il team Timing Analysis.
2. Consultare i manuali d’uso degli applicativi EDA usati da Timing Analysis.
3. Accedere alle documentazioni riguardanti i signoff e i signoff criteria.
4. Vedere gli script prodotti dal team Timing Analysis.

In figura 6.22, si può notare il diagramma della struttura di navigazione relativo a quello in figura 6.21, nel quale sono stati introdotti il menu principale “Functional Group Timing Analysis Main Menu” ed i vari index per accedere alle varie istanze di “Tool user guide”, “SignOff”, “SignOff Criteria”, “People”.

Inoltre, è stata aggiunta il query “SelectEdaToolWithUserTimingAnalysis” tramite la quale si selezionano solo le istanze di “Tool user guide” per le quali “TimingAnalysis” risulta utente.

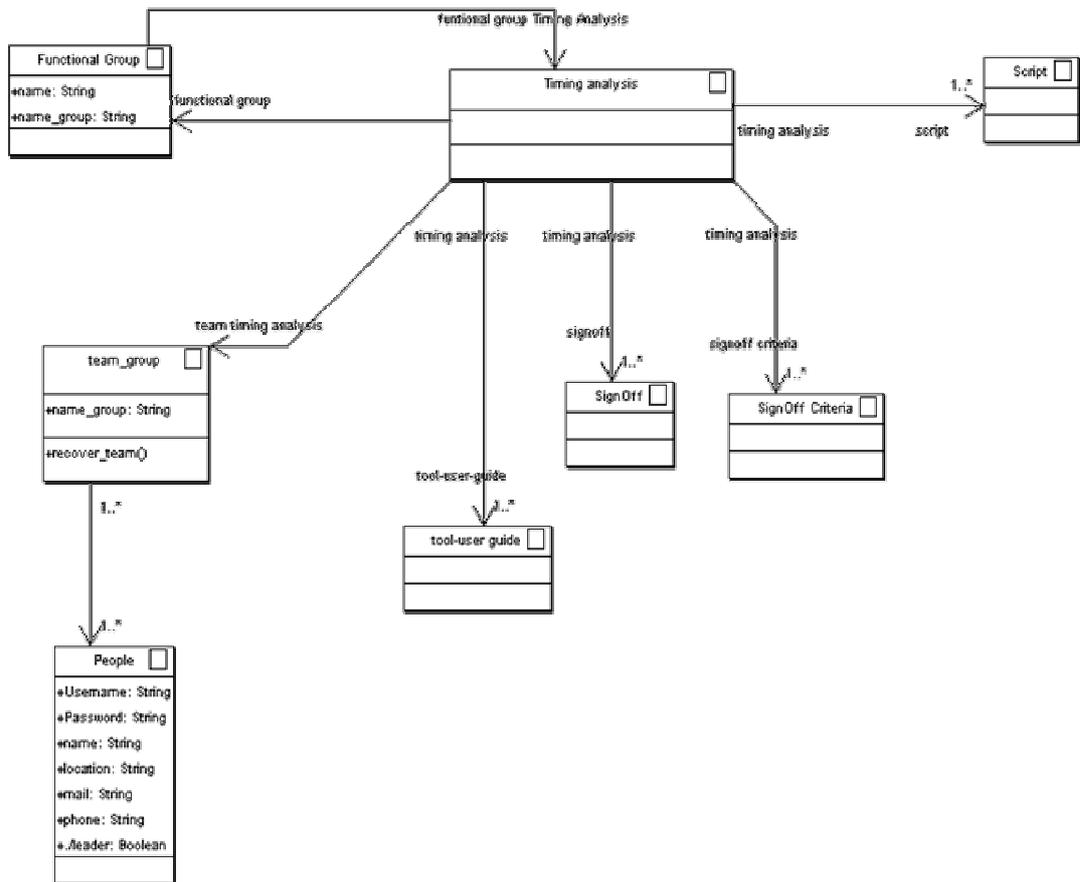


Figura 6.21 Diagramma dello spazio di navigazione Package Timing Analysis

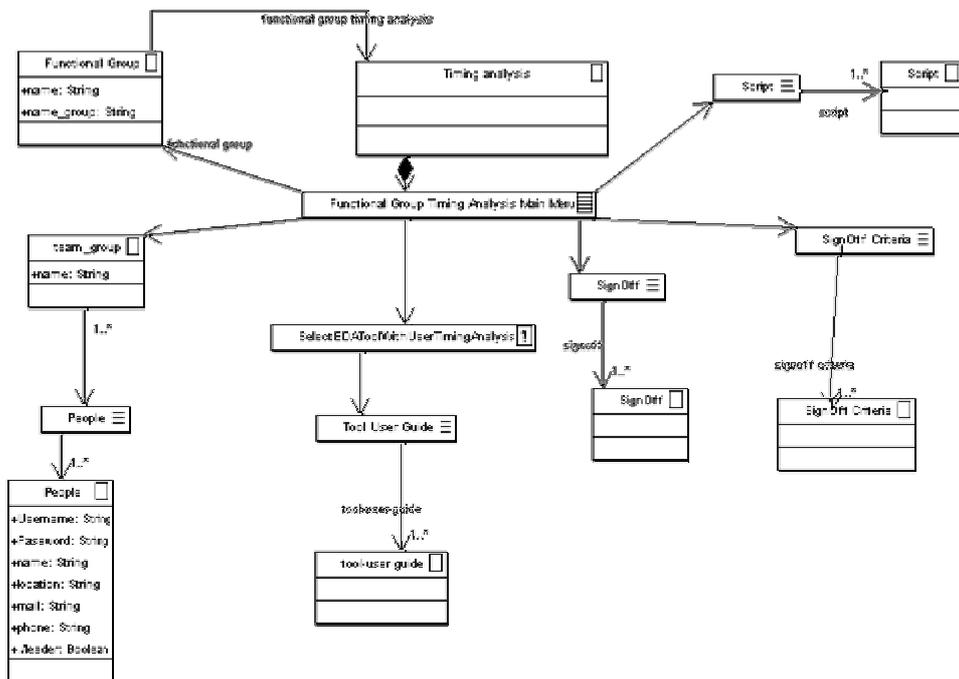


Figura 6.22 Diagramma della struttura di navigazione Package TimingAnalysis

Per il package Analog, si può notare in figura 6.23, che dall'home page del gruppo funzionale Analog possono essere effettuate le seguenti azioni:

1. Visualizzare il team Analog.
2. Consultare i manuali d'uso degli applicativi EDA usati da Analog.
3. Accedere alle documentazioni riguardanti i signoff.
4. Accedere alle documentazioni riguardanti i Technical docs, Analog Design, e progetti.

In figura 6.24, si può osservare il diagramma della struttura di navigazione relativo a quello in figura 6.23, nel quale sono stati introdotti il menu principale "Functional Group Analog Main Menu" ed i vari index per accedere alle varie istanze di "Tool user guide", "SignOff", "Technical docs", "Analog Design", "Project", "People". Inoltre, è stata aggiunta il query "SelectEdaToolWithUserAnalog" tramite il quale si selezionano solo le istanze di "Tool user guide", per le quali "Analog" risulta utente, ed il query "SelectAnalogProject" tramite il quale si selezionano solo i documenti legati ai progetti relativi alla categoria Analog.

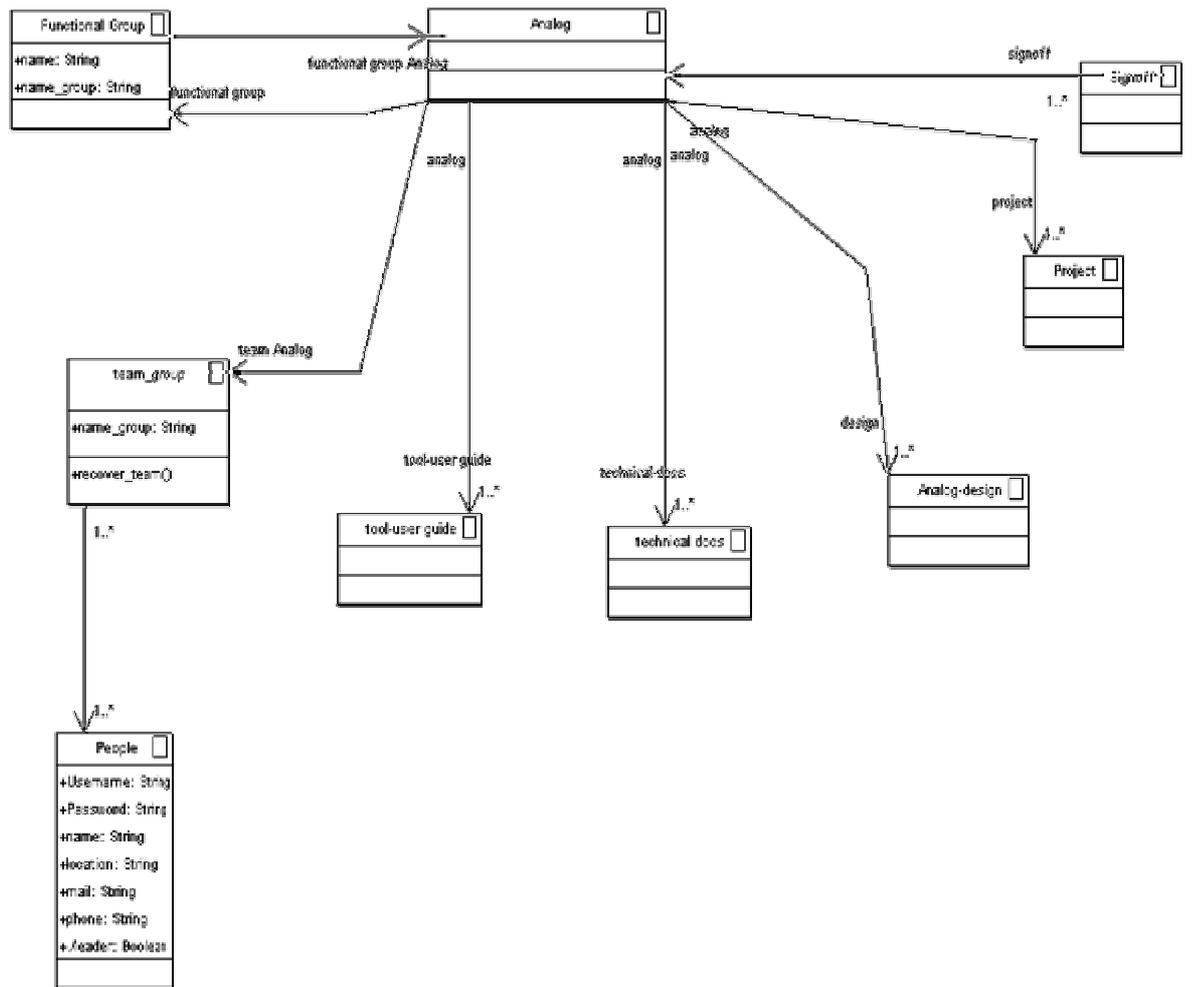


Figura 6.23 Diagramma dello spazio di navigazione Package Analog

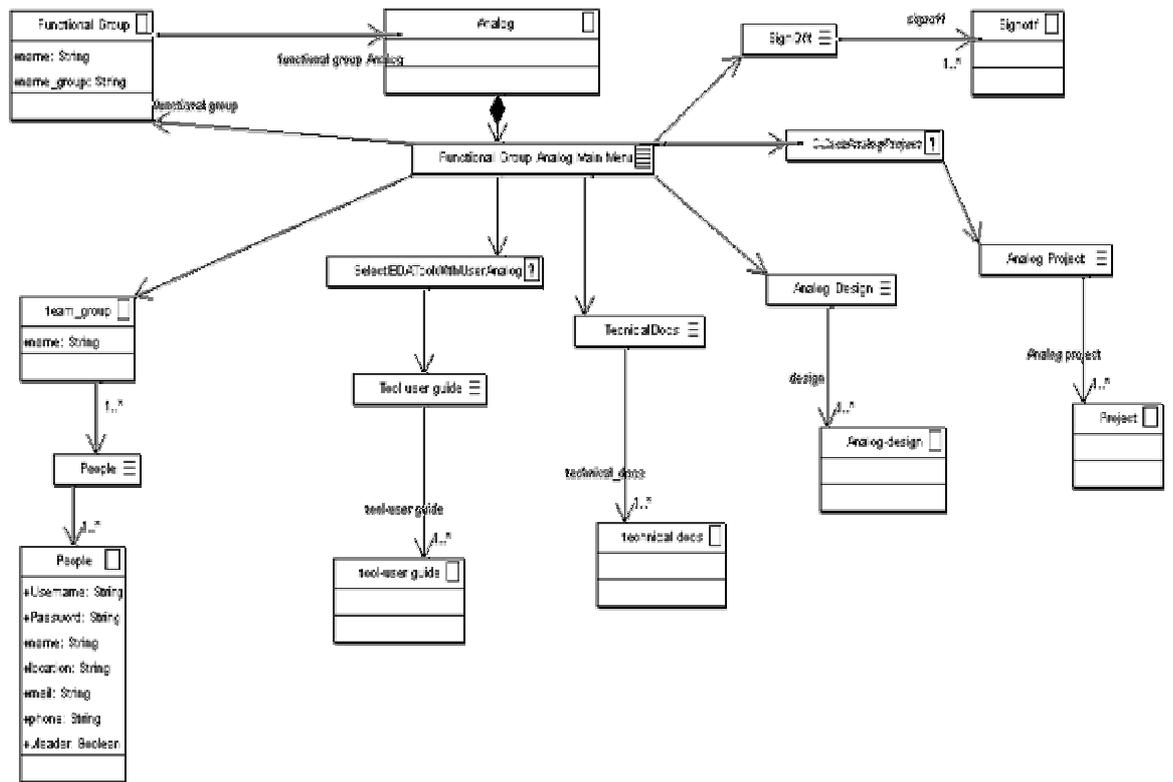


Figura 6.24 Diagramma della struttura di navigazione Package Analog

Per il package Training, si può notare in figura 6.25, che dall'home page del gruppo funzionale Training possono essere effettuate le seguenti azioni:

1. Visualizzare il team Training.
2. Consultare il catalogo dei corsi divisi per aree d'interesse.
3. Visualizzare i fornitori dei corsi, e le Training Form.

In figura 6.26, si può vedere il diagramma della struttura di navigazione relativo a quello in figura 6.25, nel quale sono stati introdotti il menu principale “Functional Group Training Main Menu” ed i vari index per accedere alle varie istanze delle classi di navigazione. Inoltre, è stata aggiunta il query “Select CatalogCourse By Family” tramite il quale si selezionano i corsi per family dove family sono le aree d'interesse.

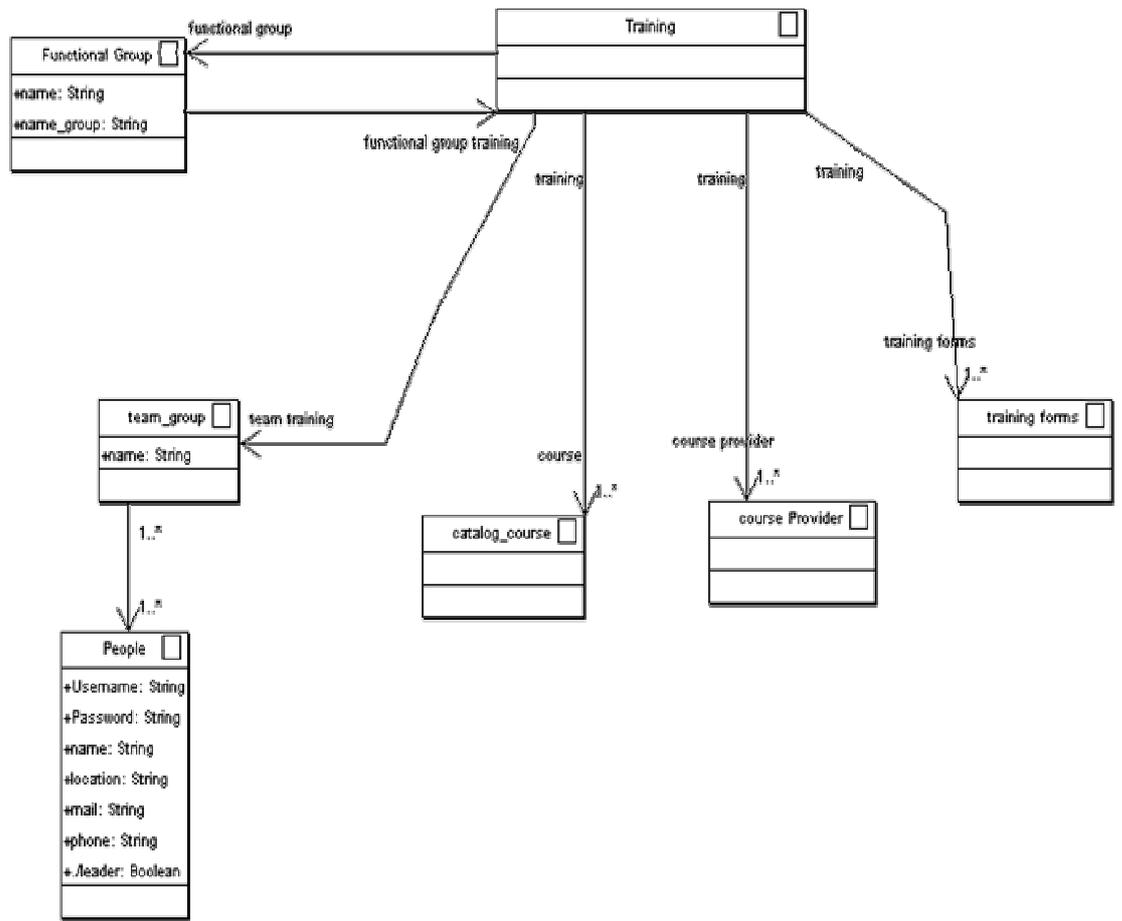


Figura 6.25 Diagramma dello spazio di navigazione Package Training

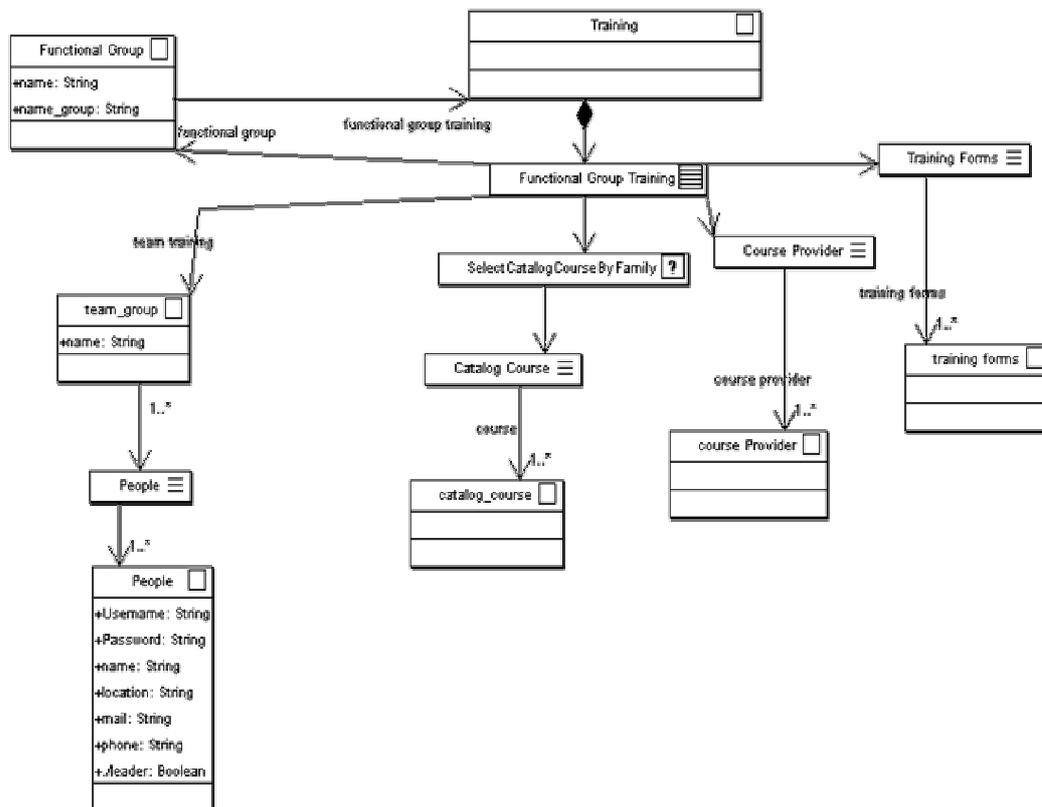


Figura 6.26 Diagramma della struttura di navigazione Package Training

In tutti i diagrammi visti sopra, si osserva la relazione biunivoca fra i vari home page e FUNCTIONAL-GROUP. Essa permette la navigazione dalla vista DVD a le varie viste dei gruppi funzionali e viceversa.

Per quanto riguarda il package “Notice Board”, lo spazio di navigazione è diverso per ogni tipo d’utente.

Se l’utente è STPeople, possiamo vedere in figura 6.27 che esso potrà accedere solo agli eventi marcati come visibili e vedere i dati dell’owner associato.

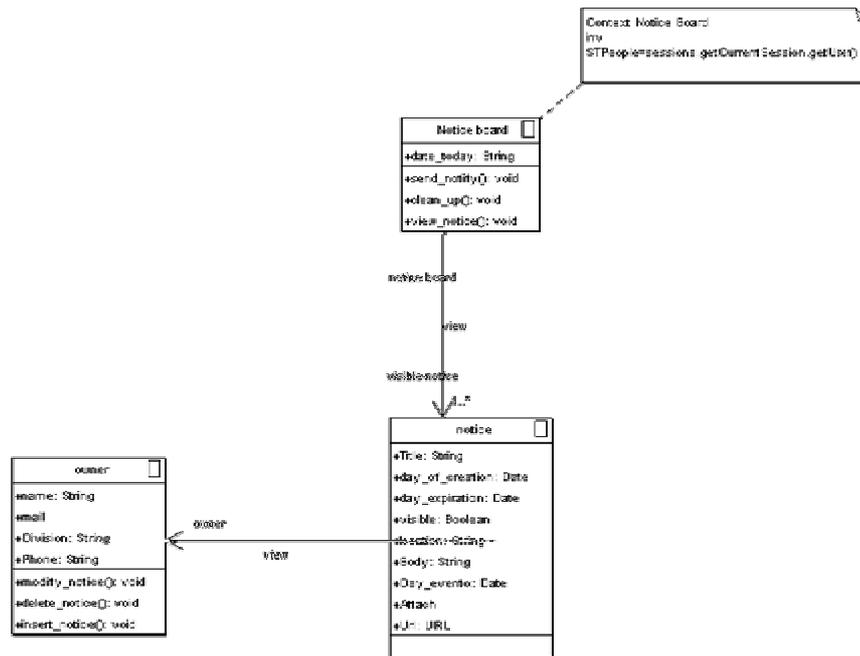


Figura 6.27 Diagramma dello spazio di navigazione Package Notice board (vista StPeople)

Mentre se l'utente è un CMGPeople, in figura 6.28, si può notare che questo tipo d'utente può vedere tutti gli eventi esistenti (visibili e non), l'owner associato, eventualmente inserire un nuovo evento.

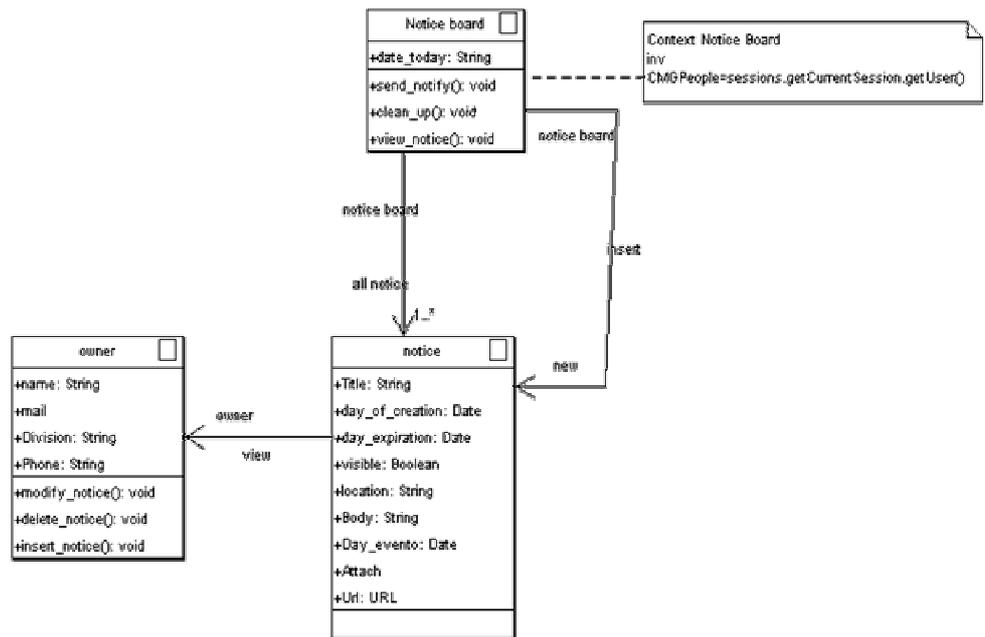


Figura 6.28 Diagramma dello spazio di navigazione Package Notice board (vista CMGPeople)

Un CMGPeople che inserisce un evento diventa owner, dal diagramma dello spazio di navigazione relativo, in figura. 6.29, si può vedere che egli potrà accedere a tutti gli eventi inseriti, potrà inserire un nuovo evento ed in più modificare o cancellare gli eventi di cui è proprietario.

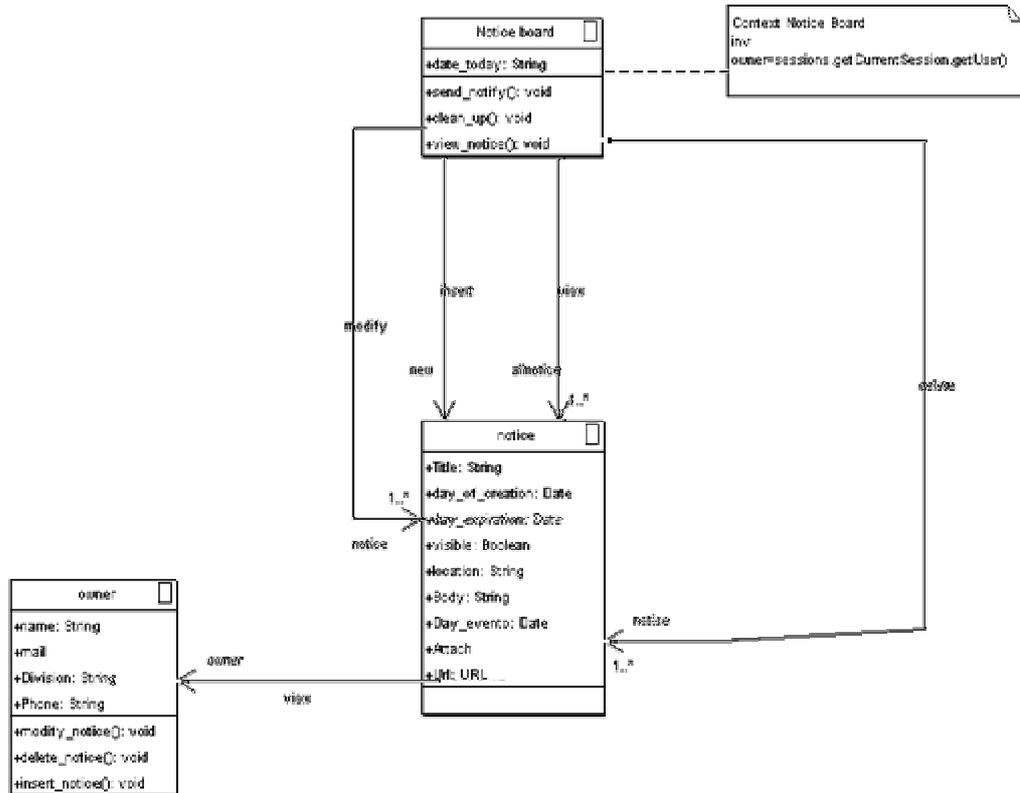


Figura 6.29 Diagramma dello spazio di navigazione Package Notice board (vista Owner)

Ai diagrammi visti nelle figure 6.27, 6.28, 6.29 sono state aggiunte le primitive d'accesso. In figura 6.30, si può vedere il diagramma della struttura di presentazione nella versione completa (vista relativa all'owner). In esso è stato inserito il menu principale dal quale si potrà accedere all'index "Notice CMG and ST", il quale consentirà di accedere a tutte le "Notice". Le query "SelectOwnerNotice" permette di estrarre dall'insieme di "Notice" quelle relative all'utente corrente. Le figure 6.31, 6.32 rappresentano i diagrammi della struttura di navigazione rispettivamente per STPeople e CMGPeople.

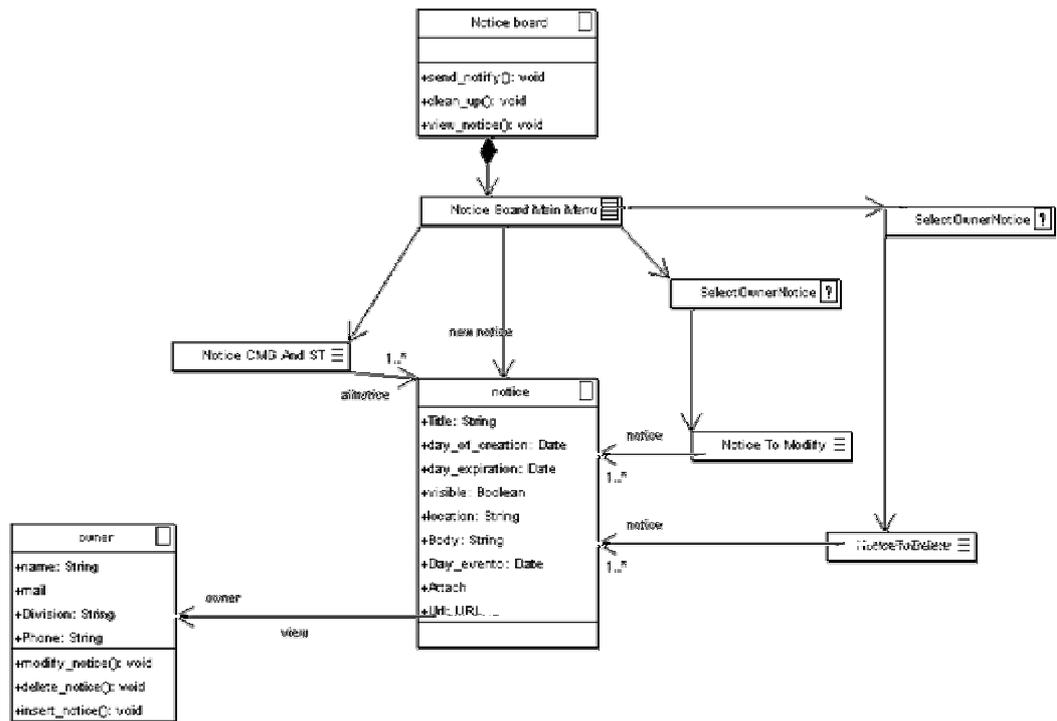


Figura 6.30 Diagramma della struttura di navigazione Package Notice board(vista owner)

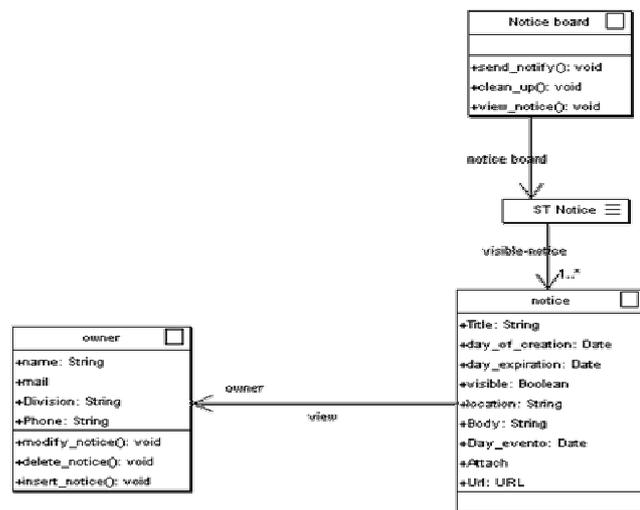


Figura 6.31. Diagramma della struttura di navigazione Package Notice board (vista STPeople)

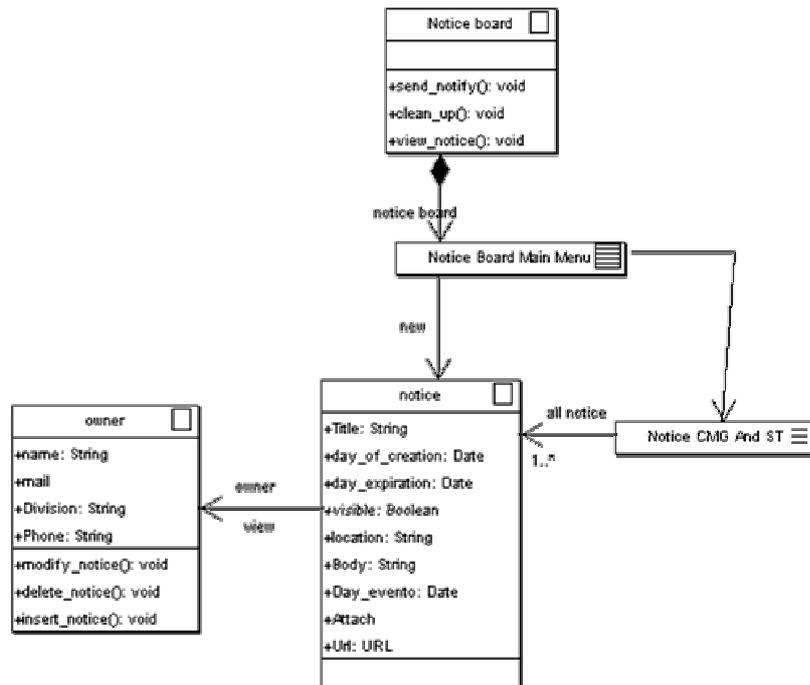


Figura 6.32. Diagramma della struttura di navigazione Package Notice board (vista CMGPeople)

5.3 Diagramma di Presentazione

Secondo la metodologia mostrata nel capitolo 1, il modello di presentazione modella gli aspetti statici e dinamici delle presentazioni di un'applicazione Web.

Purtroppo, però, il tool attualmente disponibile per la metodologia UWE, non supporta la modellazione degli aspetti dinamici della presentazione così come è stato detto nel capitolo 3.

In questo paragrafo quindi verranno mostrati solo gli aspetti di presentazione a livello statico. Gli aspetti dinamici sono stati concordati direttamente con i responsabili delle divisioni DVD e STB, per esse non è stata prodotta però una documentazione scritta.

Al fine di soddisfare gli standard aziendali per la rappresentazione di pagine Web interdivisionali, è stato utilizzato per l'interfaccia (quindi per l'implementazione degli aspetti di presentazione) il *template* fornito dalla ST, esso suggerisce una presentazione di tipo menu-based, a finestra singola e senza frame.

Nei diagrammi sottostanti, si possono vedere le classi di presentazione create per ogni classe di navigazione e per ogni attributo delle classi di navigazione.

Le classi di presentazione per gli attributi sono connesse alle classi di presentazione della classe di navigazione a cui appartengono, tramite composizione.

Per ogni menu viene creata una classe di presentazione, legata tramite composizione con la classe di navigazione che contiene il menu. Ad ogni "index" e "query" viene associata una classe di presentazione. In particolare, se un index od un query è puntato direttamente da un menù esso avrà una relazione di composizione con il menu per indicare che la pagina relativa sarà composta da un menu nella parte sinistra, e le relative presentazioni di index o query nella parte destra tramite le quali si accede alle classi di navigazioni, il cui contenuto è presentato dalla relativa classe di presentazione.

Diagramma di presentazione per il Package DVD:

Le seguenti classi di presentazione sono derivate dal diagramma della struttura di presentazione mostrato in figura 6.17.

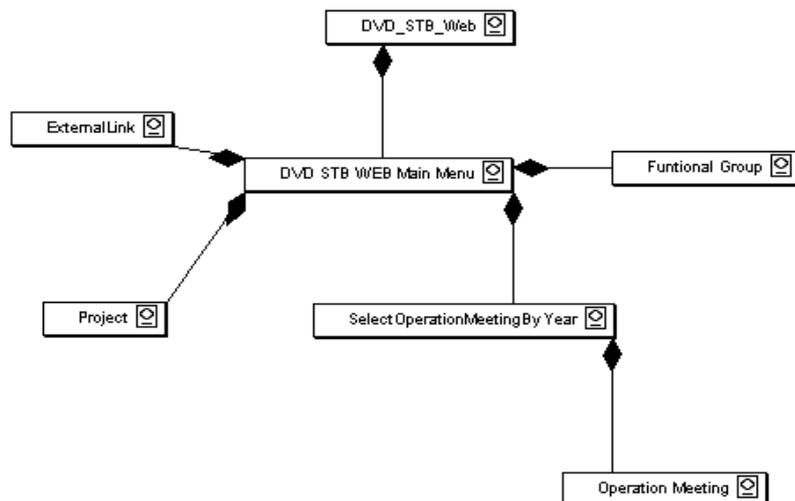


Figura 6.33. Classe di presentazione DVD_STB_Web

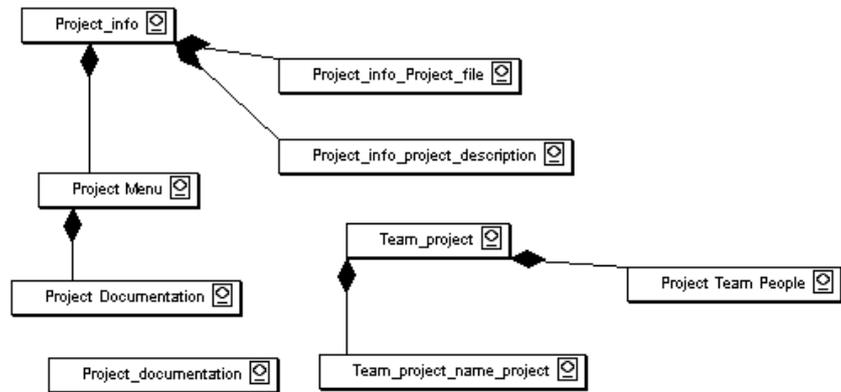


Figura 6.34. Classi di presentazione Project_info e Team_project

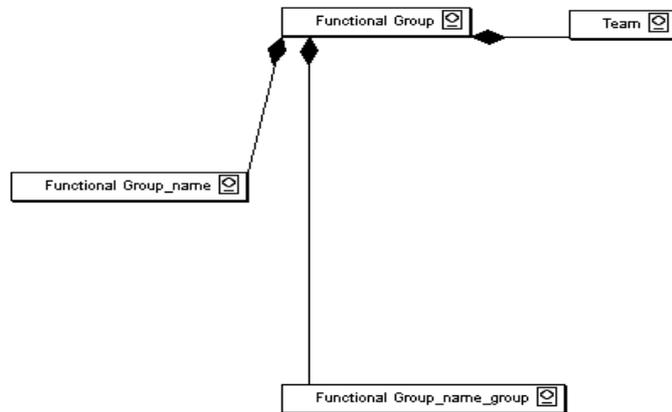


Figura 6.35. Classe di presentazione Functional Group

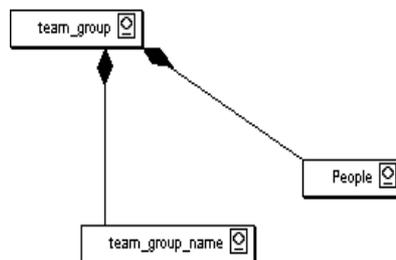


Figura 6.36. Classe di presentazione Team Group

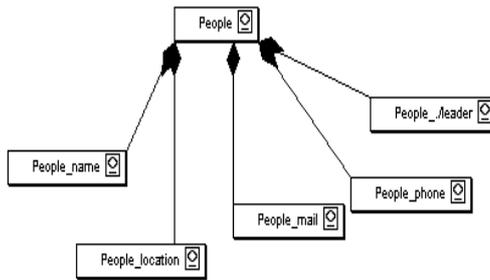


Figura 6.37. Classe di presentazione People

Diagramma di presentazione Package DVD-Vista Team leader:

Le seguenti classi di presentazione sono derivate dal diagramma della struttura di presentazione mostrato in figura 6.18.

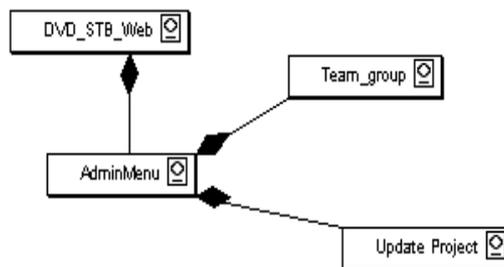


Figura 6.38. Classe di presentazione DVD_STB_Web e AdminMenu

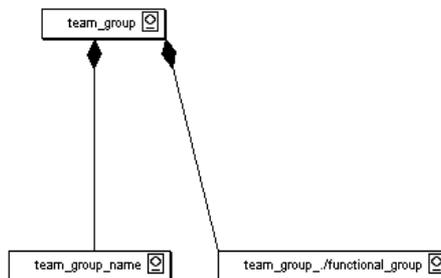


Figura 6.39. Classe di presentazione TeamGroup

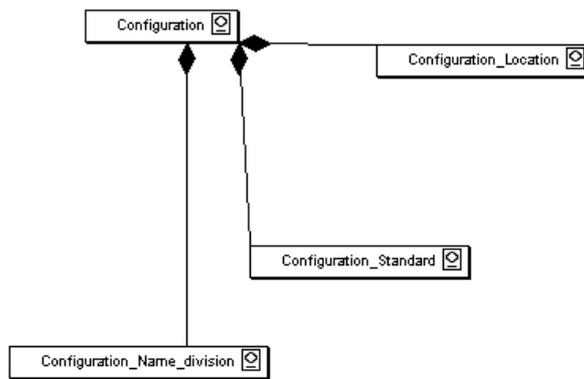


Figura 6.40. Classe di presentazione Configuration

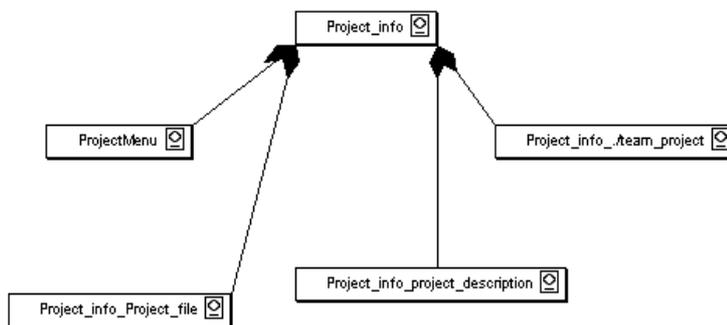


Figura 6.41. Classe di presentazione Project_Info

Diagramma di presentazione Package Analog:

Qui sono mostrate le classi di presentazione principali relative al diagramma della struttura di presentazione mostrato in figura 6.24.

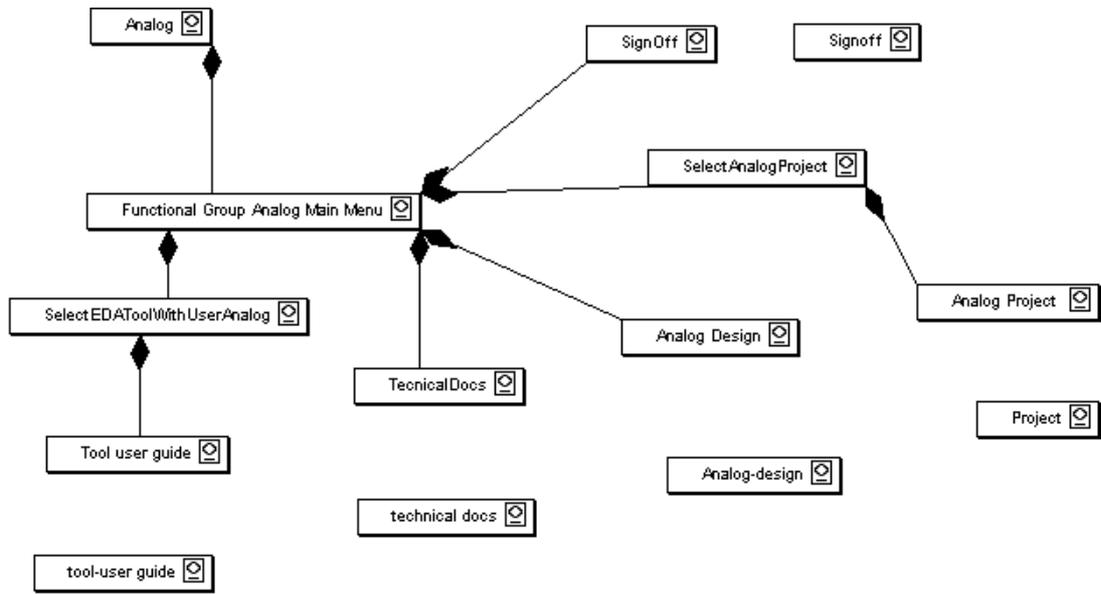


Figura 6.42. Classi di presentazione Analog, Tool-user guide, Technical docs, Analog Design, Project, SignOff.

Diagramma di presentazione Package Timing Analysis:

Classi di presentazione principali relative al diagramma della struttura di presentazione mostrato in figura 6.22.

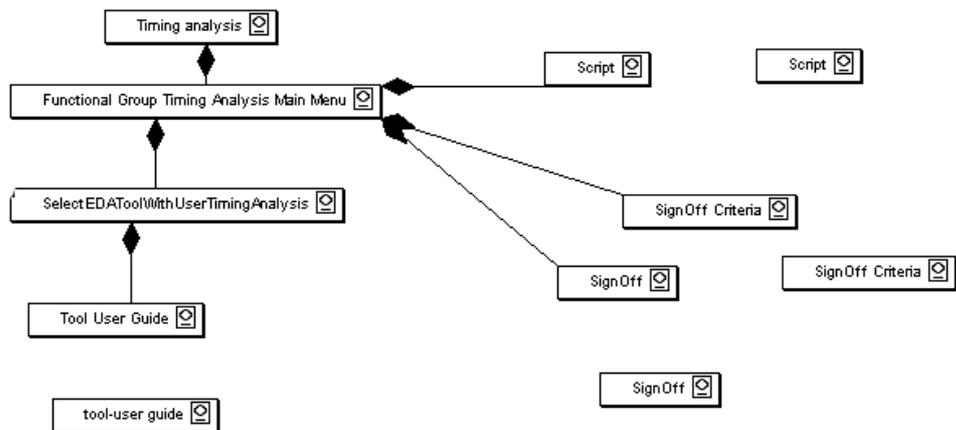


Figura 6.43. Classi di presentazione Timing analysis, tool-user guide, SignOff, SignOff Criteria, Script.

Diagramma di presentazione Package EDA:

In questa vista sono mostrate solo le classi di presentazione principali relative al diagramma della struttura di presentazione mostrato in figura 6.20.

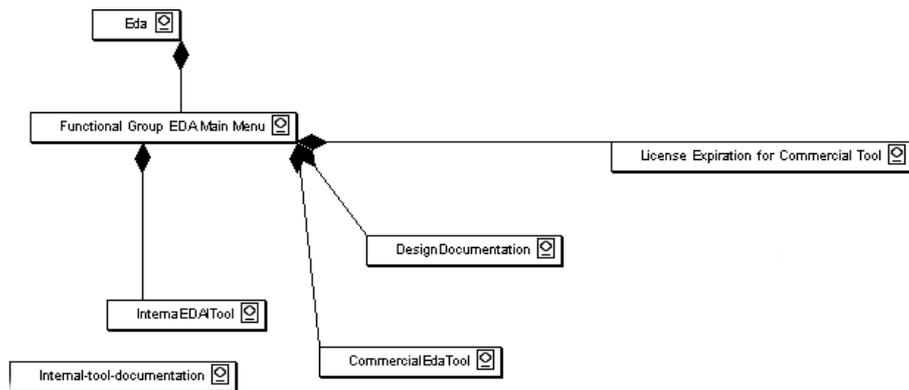


Figura 6.44. Classi di presentazione EDA, Internal_tool_documentation.

Diagramma di presentazione Package Training:

Qui sono mostrate le classi di presentazione principali relative al diagramma della struttura di presentazione mostrato in figura 6.26

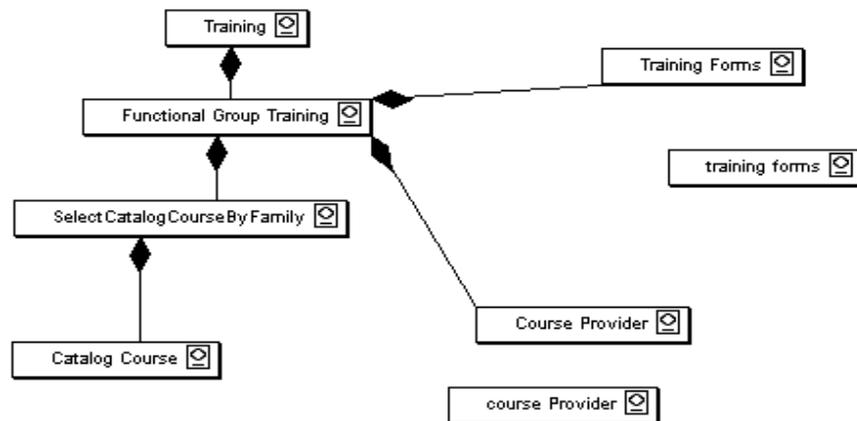


Figura 6.45. Classi di presentazione Training, Course Provider, training forms

Diagramma di presentazione Package Notice Board (vista Owner):

Per quanto riguarda le classi relative al package Notice Board mostreremo solo quelle relative alla vista dell'owner, perché le classi per le altre viste (CMGPeople e STPeople sono derivate da essa).

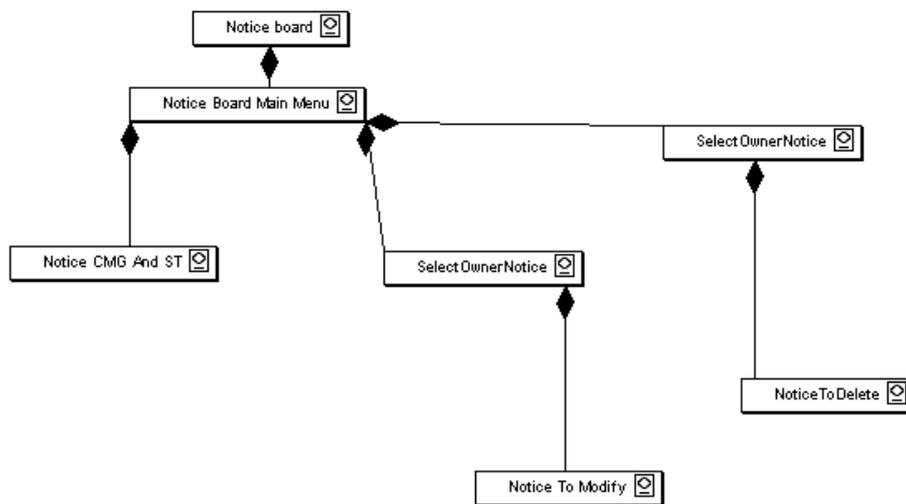


Figura 6.46. Classi di presentazione Notice board.

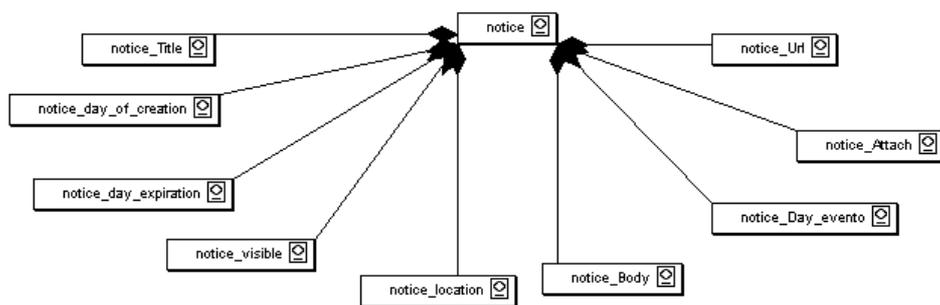


Figura 6.47. Classi di presentazione notice.

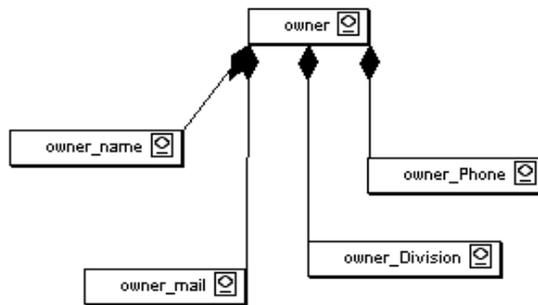


Figura 6.48. Classi di presentazione owner.

6 Progettazione della Struttura dell' applicazione

Uno schema di come sarà strutturata l'architettura di DVD-STB Web è mostrata in figura 6.49.

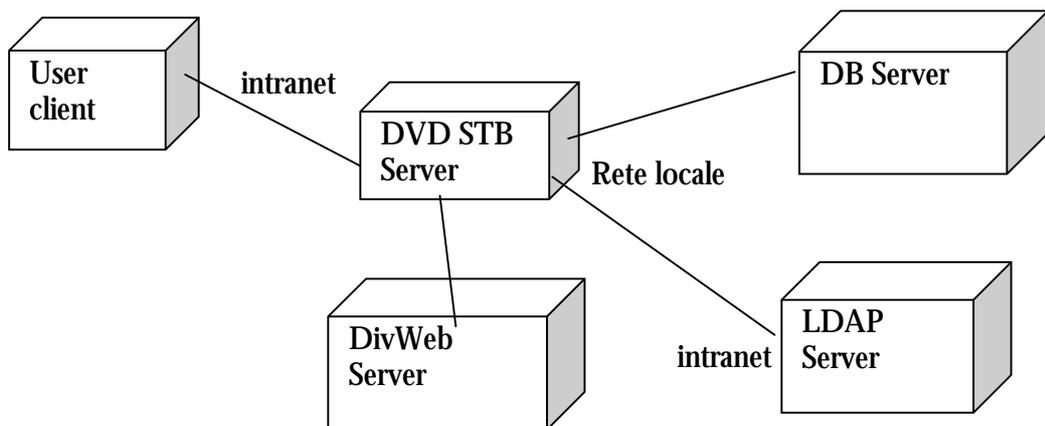


Figura 6.49 Struttura dell'applicazione DVD_STB WEB

6.1 Progettazione dettagliata delle classi

Come detto nel capitolo 5, nella fase di analisi e design sono stati prodotti diversi diagrammi: concettuale, di navigazione e di presentazione. In questi diagrammi è stato introdotto un insieme di classi. In quella fase vengono identificate: le classi, qualche attributo, e a volte anche qualche operazione.

Nelle fasi successive, le classi sono arricchite da qualche dettaglio. Questa attività include le seguenti sotto attività: definire le operazioni, definire gli attributi, identificare le aggregazioni, le associazioni, l'ereditarietà e le dipendenze fra classi, descrivere i metodi, determinare gli stati e stabilire i requisiti rilevanti per la loro implementazione.

Per dare una descrizione dettagliata delle classi introdotte, è necessario seguire il procedimento nel capitolo 5 nel paragrafo “Progettazione dettagliata delle classi”, la descrizione di tutte le classi esula dagli scopi di questa tesi. Questo tipo di documentazione sarà prodotta in seguito, per rilasciare alla STMicroelectronics, la documentazione completa dell'intero prodotto da me realizzato.

6.2 Definizione di sottosistemi e loro interfacce

I sottosistemi individuati sono rappresentati in figura.

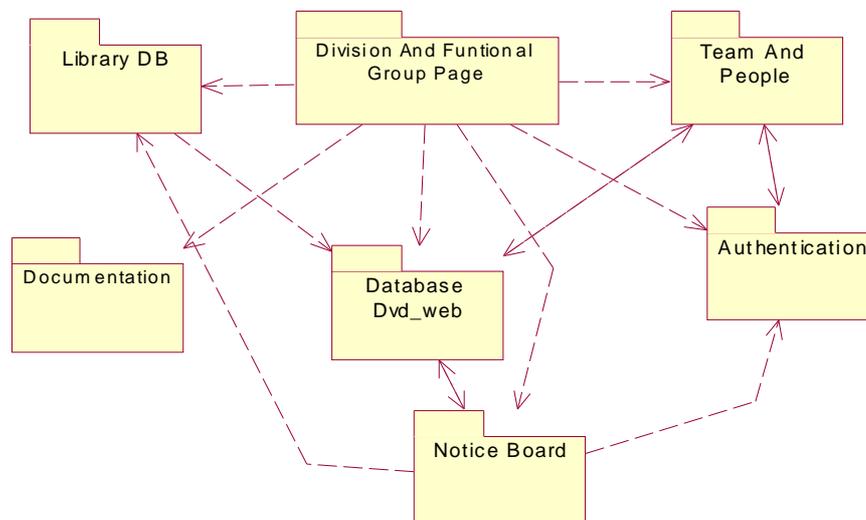


Figura 6.50. Sottosistemi di DVD STB Web

7 Implementazione

La fase d'implementazione consiste nella trasformazione dei risultati prodotti nella fase di design (classi, sottosistemi ed interfacce, ecc), in un sistema implementato in termini di componenti (codice, script, eseguibili, ecc.).

Come detto nel capitolo 4, per l'implementazione si utilizza il linguaggio script PHP4, per gli script lato server, mentre come RDBMS si usa mysql.

7.1 Diagramma dei Componenti

In figura 6.51 si può vedere il diagramma dei componenti vista generale.

Il package “DVD-STB Web” indica il directory in cui è contenuta l'applicazione nel Web Server. Mentre “DVD_Web” è il database creato per l'applicazione; esso è contenuto fisicamente nel database server, le sue tabelle implementano le classi concettuali omonime. “Member” è stata aggiunta per assicurare l'associazione fra le istanze della classe “people” e “team_group”, e “people” e “team_project”. Per quanto riguarda i progetti le informazioni relative alla classe “team_project” sono state incluse in “Project_info”.

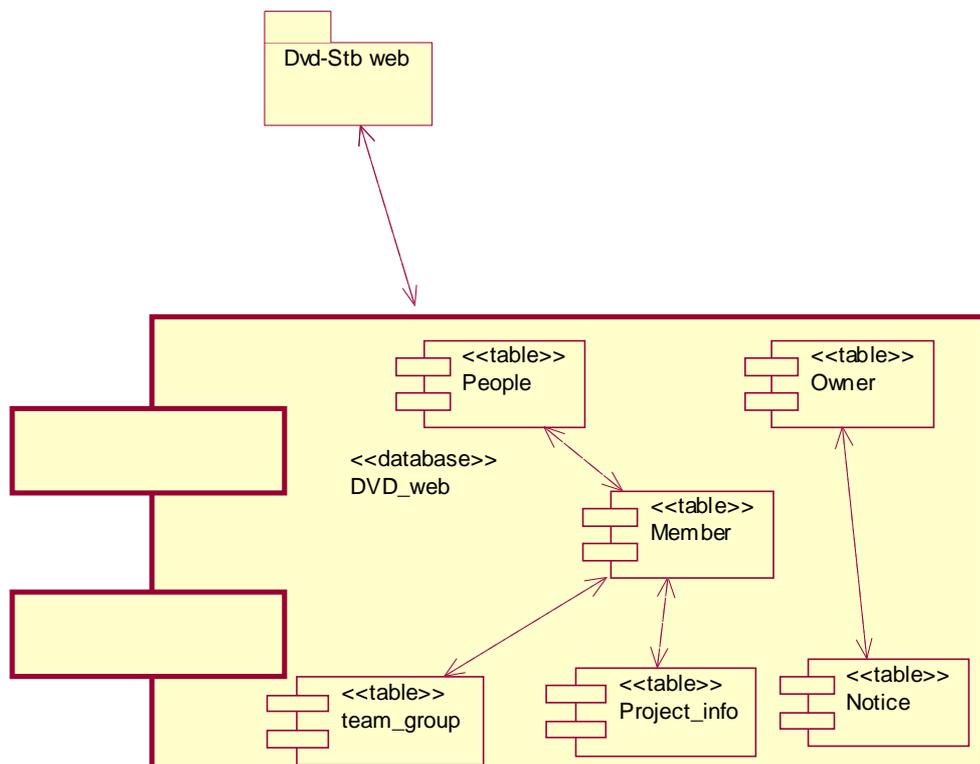


Figura 6.51. Diagramma dei componenti

Il package “DVD_STB Web” al suo interno è strutturato come in figura 6.52.

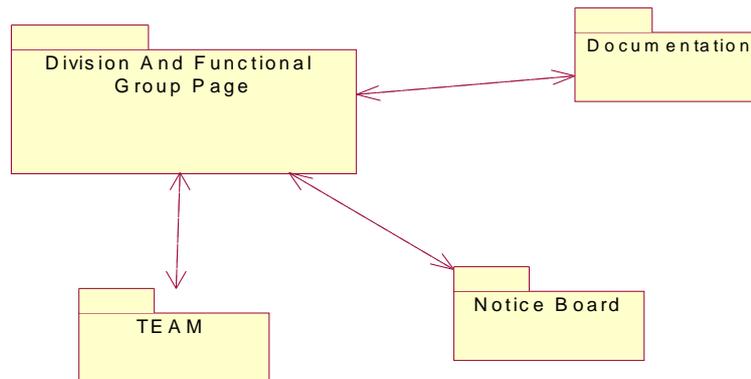


Figura 6.52. Diagramma dei componenti (Struttura del Package DVD-STB Web)

Il package (o directory) “Division And Functional Group Page” contiene tutti i file e gli script necessari per creare la struttura delle home della divisione e dei gruppi funzionali. Il diagramma relativo è rappresentato in figura 6.53.

“Dvd-Web.php”, “Eda.php”, “Analog.php”, “Training.php”, “Functional_group.php” implementano le classi di presentazione omonime. “Filehtm.inc” è un file che include tutte le funzioni necessarie per creare le pagine html. “Funzioni_comuni.inc” e “common_db.inc” contengono funzioni e configurazioni necessarie per interrogare i database. “Coommon_ldap.inc” e “Function_authenticate.inc” contengono le configurazioni e le funzioni necessarie per autenticare un utente tramite l’Enterprise Directory.

“New_update_project.php” permette di creare un nuovo progetto ed aggiornarlo, tramite esso si creano istanze di “project_info”.

“Config_dvd_stb.php” permette di cambiare le configurazioni, ed i nomi dei team da recuperare.

“view_people.php” implementa la classe di presentazione “People”.

In figura 6.54 è mostrato il contenuto del package “Documentation”. Esso permette di creare tutte le classi di presentazione necessarie per le classi che sono derivate dalla tabella “DVD” del database dell'applicazione Div_Web. (Si veda il paragrafo analisi e design dell'applicazione DVD-STB Web).

In figura 6.55 si possono vedere tutti i file necessari per recuperare visualizzare ed aggiornare i componenti dei vari team.

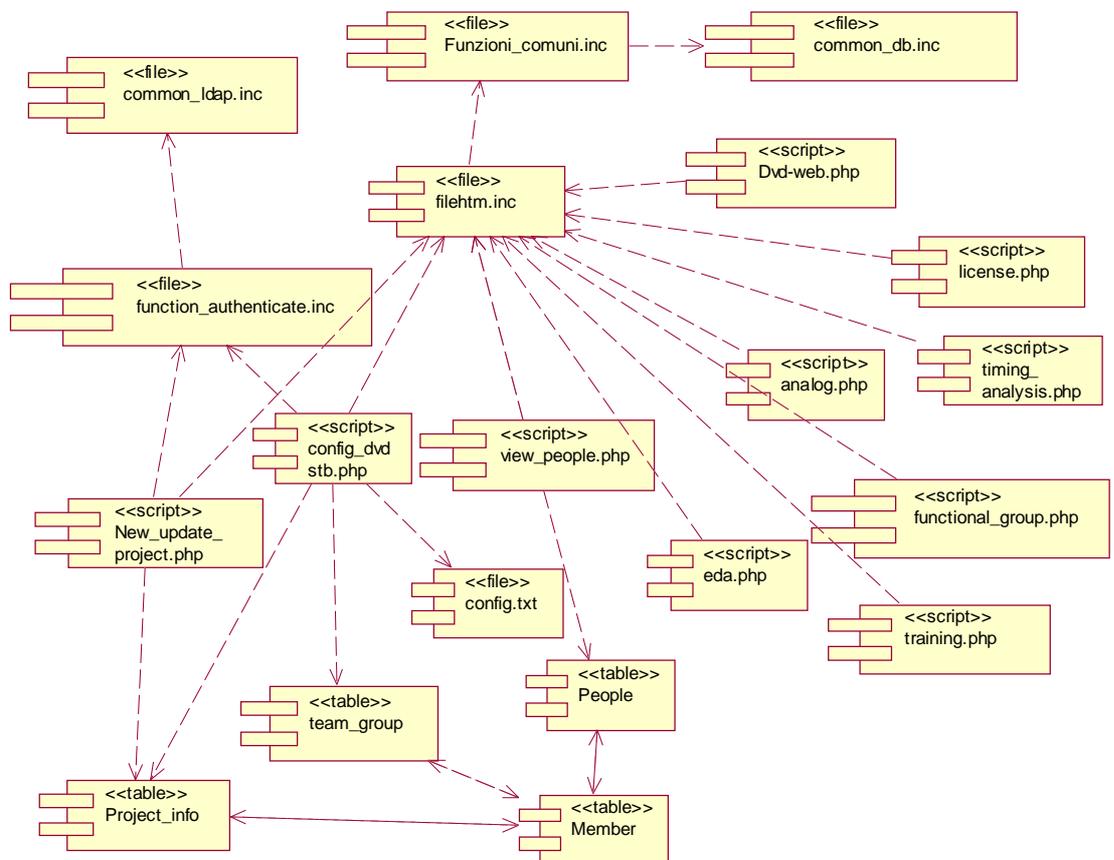


Figura 6.53. Diagramma dei componenti (Struttura del Package Division and Functional Group Page)

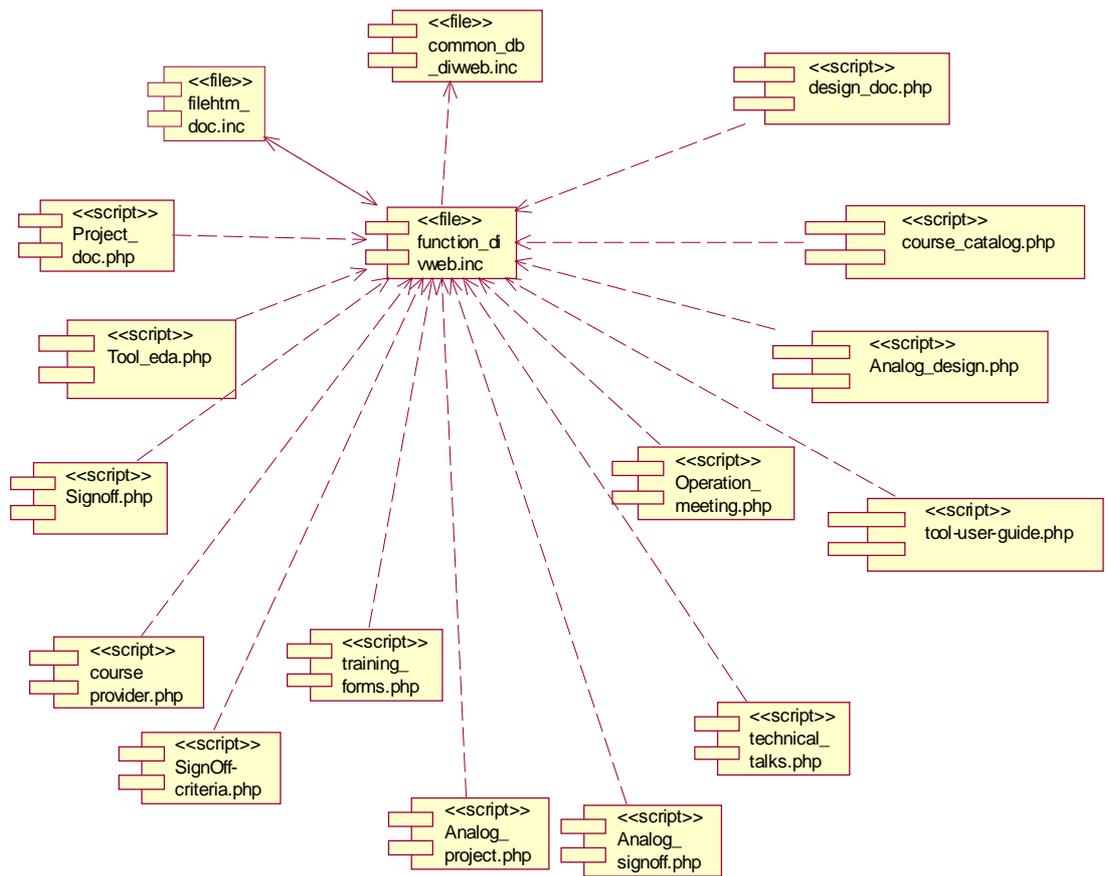


Figura 6.54. Diagramma dei componenti (Struttura del Package Documentation)

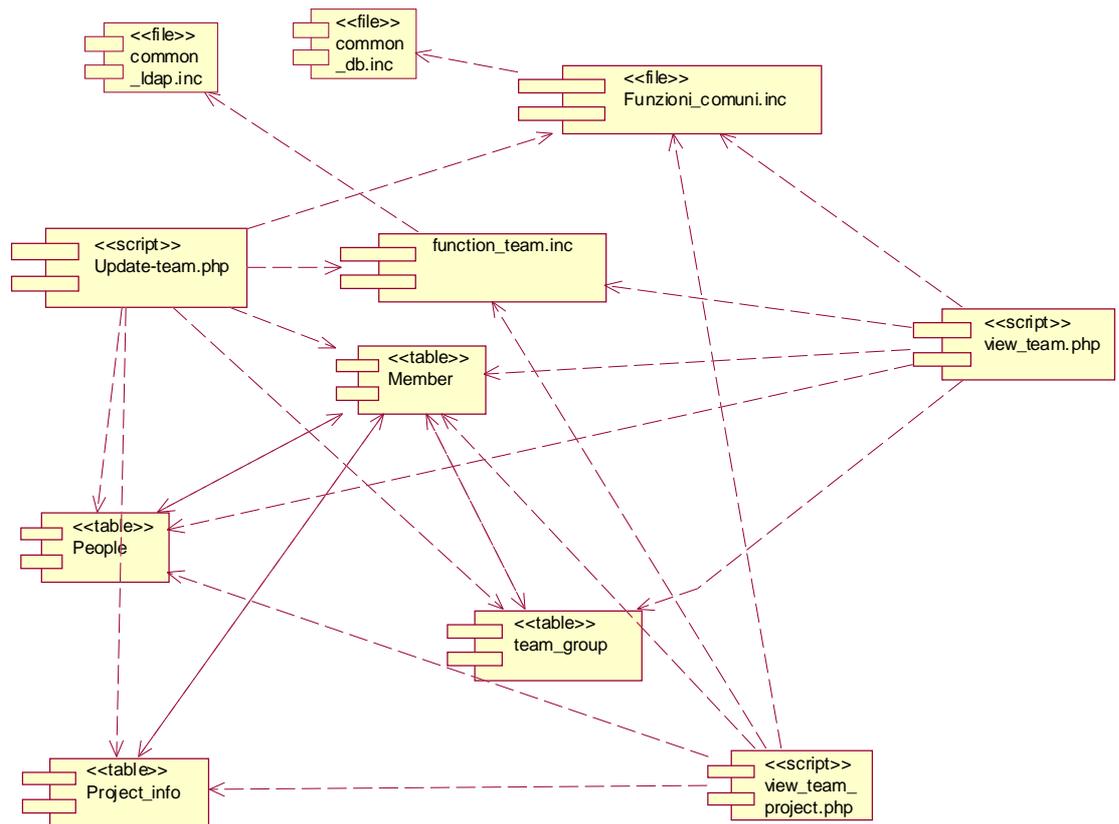


Figura 6.55. Diagramma dei componenti (Struttura del Package Team)

In figura 6.56 sono sostenuti tutti i file necessari per implementare la bacheca elettronica. In particolare "notice_board.php" implementa la classe di presentazione omonima, mentre "modify_notice.php", "delete_notice.php", "view_notice.php" rispettivamente rendono possibile la modifica, cancellazione e visione dei "notice".

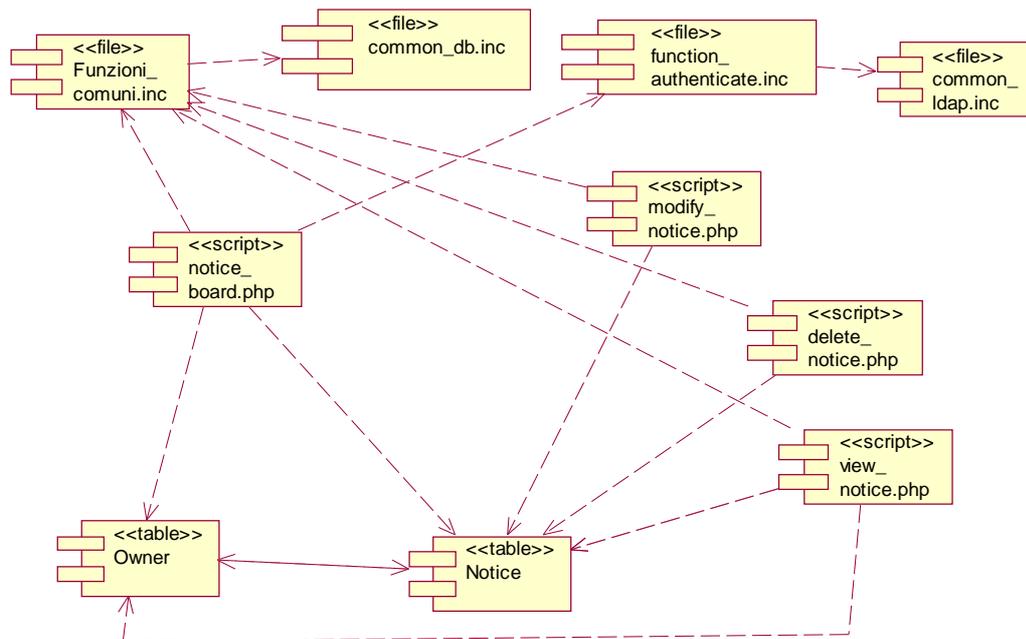


Figura 6.56. Diagramma dei componenti (Struttura del Package Notice Board)

7.2 Diagramma di Dispiegamento

I diagrammi di dispiegamento o *deployment diagram* mostrano l'architettura hardware e software del sistema. Nel diagramma in figura 6.57 si può notare come sono interconnessi fra di loro gli elementi fisici dell'applicazione in questione.

I componenti analizzati precedentemente saranno localizzati come segue:

- Il database DB DVD_Web visto in figura 6.51 fisicamente è localizzato nel database server.
- Il package “DVD_STB Web” è fisicamente presente nel Web server.
- “Enterprise Directory” e “The Group Service” si trovano fisicamente nello ldap server.
- “Div-Web” ed il suo database si trovano nel div-web server.

Il client otterrà tutte le informazioni elaborate dal Web server tramite il protocollo http.

La comunicazione fra il Web server, database server e div-web server (database di Div_Web) il protocollo di comunicazione sarà il TCP/IP.

Fra Web server e ldap server il protocollo di comunicazione sarà proprio ldap.

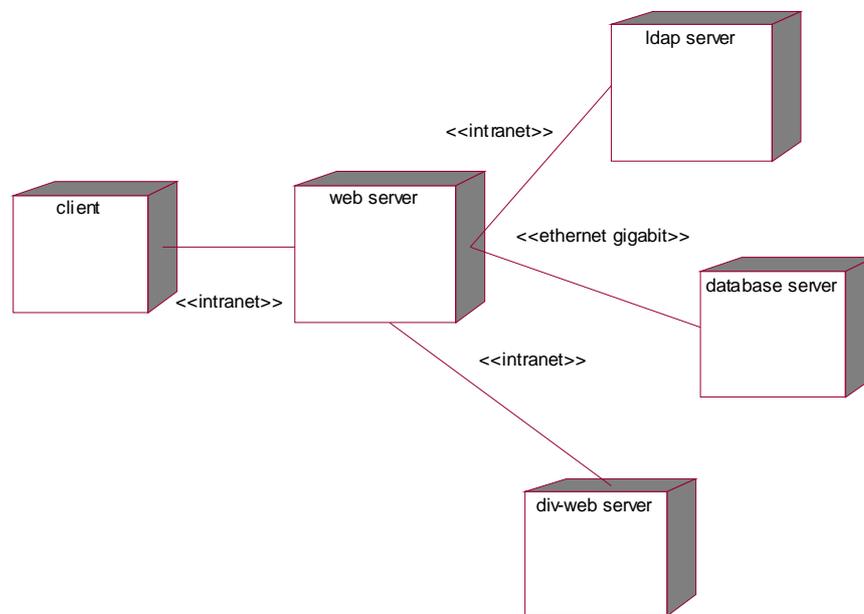


Figura 6.54. Diagramma di dispiegamento DVD STB Web

C o n c l u s i o n i

L'utilizzo della metodologia scelta non ha deluso le aspettative, anzi, è risultata molto utile per la progettazione delle applicazioni in studio. Essa, infatti, ha permesso di strutturare il lavoro in fasi considerando i vari aspetti di un'applicazione Web. Consentendo, cioè, di analizzare, innanzitutto, gli elementi del dominio, quindi gli aspetti legati alla navigazione ed, infine, trattare gli aspetti di presentazione, parte molto importante per un'applicazione Web. Tramite questa divisione del lavoro è immediato ottenere un'implementazione semplificata strutturando al meglio il carico di lavoro.

In altri termini, la metodologia dello UWE rappresenta un valido supporto per la definizione della struttura dei database ed altre classi necessarie per l'implementazione, ed anche per la definizione dei cammini di navigazione consentendo di considerare il miglior cammino da seguire, in tal modo si evita di sovraccaricare le applicazioni con cammini non ottimizzati. Grazie ai diagrammi della struttura di navigazione è possibile conoscere a priori la struttura delle pagine Web e l'allocazione dei menu e delle primitive d'accesso.

Tutto ciò ha aiutato da un lato il cliente a capire meglio le proprie esigenze e, quindi meglio definire le specifiche e dall'altro, nel caso in cui ci fossero stati più implementatori, sicuramente avrebbe fornito un valido strumento per la distribuzione del lavoro e, per tanto, per l'implementazione del progetto.

I limiti del tool, però, sia per quanto riguarda la progettazione della struttura di navigazione che per la progettazione dei dettagli relativi alla presentazione, sono notevoli. Bloccano il flusso di modellazione, perché nel primo caso non è possibile inserire sottomenu, o avere dettagli più concreti su "IndexItem" e "guidedTour", mentre per quanto riguarda gli aspetti di presentazione, al momento, non è possibile effettuare una loro modellazione concreta sia a livello statico che dinamico.

Il limite relativo alla mancanza della modellazione degli aspetti di presentazione è stato molto sentito anche dal cliente, il quale più volte ha espresso il suo desiderio di vedere una struttura della presentazione formale prima che si passasse all'implementazione del progetto.

Questa modellazione risulterebbe molto interessante perché non solo faciliterebbe la comunicazione con il cliente, ma per di più permetterebbe una verifica delle specifiche ancor prima della fase implementativa e migliorerebbe quest'ultima rendendola ancor più strutturata, lineare e mirata.

ESTENSIONE UML PER APPLICAZIONI WEB

Questa appendice presenta il profilo UML per applicazioni Web. Gli stereotipi sono stati definiti e spiegati dettagliatamente nel capitolo 1. Quest'estensione è basata sul meccanismo d'estensione generale fornito da UML. L'estensione include degli stereotipi specifici per modellare la navigazione, presentazione ed aspetti adattativi delle applicazioni multimediali. I modelli che usufruiscono dell'estensione UML sono: il modello dello spazio di navigazione, il modello della struttura di navigazione, il modello adattativi, ed i modelli di presentazione. Solo pochi stereotipi sono specifici della modellazione di caratteristiche adattative. Gli stereotipi, sono quindi raggruppati in stereotipi per applicazioni multimediali generiche e stereotipi specifici per applicazioni multimediali adattative. Per gli elementi standard UML e gli stereotipi standard si fa riferimento a [18].

1 Stereotipi per modellare applicazioni multimediali generiche

Il profilo UML definisce i seguenti stereotipi per la progettazione di applicazioni multimediali generiche.

STEREOTIPI/ICONE	APPLICATI A	SIGNIFICATO
Ancor 	Classi	indica una classe, nella quale i suoi oggetti sono delle aree cliccabili, che hanno dei link associati ad altri nodi
Anchored collection 	Classi	Indica una classe, nella quale i suoi oggetti sono delle liste di ancore
Audio 	classi	Indica una classe, nella quale i suoi oggetti sono delle sequenze audio che possono essere riprodotte, fermate, ecc.
button 	classi	Indica una classe, nella quale i suoi oggetti sono delle aree cliccabili, con delle azioni associate ad esse.

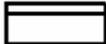
STEREOTIPI/ICONE	APPLICATI A	SIGNIFICATO
Collection 	classi	Indica una classe, nella quale i suoi oggetti sono un insieme di altri elementi, come testo, immagini, ecc. Non è specificato come l'insieme verrà visualizzato.
Direct navigability	associazioni	Indica che l'oggetto puntato dall'associazione è accessibile tramite una navigazione diretta dall'oggetto sorgente. La direzione dell'associazione è indicata da una freccia che è attaccata in uno od in entrambi gli estremi dell'associazione (bi direzionale).
External node 	classi	Indica una classe, nella quale i suoi oggetti sono degli obiettivi di navigazione che però sono il risultato di un'altra applicazione Web.
Form 	classi	Indica una classe, nella quale i suoi oggetti sono usati per richiedere informazioni.
Frame 	classi	Indica la porzione di più basso livello in cui un frameset è diviso.
Frameset 	classi	Indica una classe, nella quale i suoi oggetti sono elementi di più alto livello modellati tramite una composizione, i quali contengono oggetti di più basso livello (i frame). I frameset appartengono sempre ad una finestra e possono essere eventualmente nidificati.

STEREOTIPI/ICONE	APPLICATI A	SIGNIFICATO
Guided tour 	classi	Indica una classe, nella quale i suoi oggetti forniscono un accesso sequenziale ad istanze di una classe di navigazione. L'associazione diretta che connette un guided tour con una classe di navigazione ha l'etichetta {ordered}.
Image 	Classi	Indica una classe, nella quale le sue istanze sono degli oggetti multimediali visualizzabili.
Index 	Classi	Indica una classe formata da oggetti composti che contengono un numero arbitrario di index item. Ogni index item è a sua volta un oggetto con un nome ed un link ad una classe di navigazione.
Menu 	Classi	Indica una classe formata da oggetti composti che contengono un numero fissato di menu item. Ogni menu item contiene un nome costante ed un link, che può puntare o ad una classe di navigazione o ad una primitiva d'accesso.
Navigation 	Classi	Indica una classe, nella quale le sue istanze sono ottenute da un oggetto concettuale corrispondente e sono visitati dall'utente durante la navigazione.
Presentation 	Classi	Indica una classe, nella quale le sue istanze costituiscono la presentazione d'oggetti di navigazione o di primitive d'accesso. E' un contenitore che contiene elementi, come testo, immagini, ancore, ecc.

STEREOTIPI/ICONE	APPLICATI A	SIGNIFICATO
Presents	Associazioni	Indica che l'oggetto che viene puntato da quest'associazione sarà visualizzato nella locazione indicata dall'oggetto dal quale è nata l'associazione.
Query 	Classi	Indica una classe, nella quale i suoi oggetti hanno una stringa di query come attributo. Queste stringhe possono essere delle operazioni di selezione OCL.
Text 	Classi	Indica una classe, nella quale i suoi oggetti sono una sequenza di caratteri.
Video 	Classi	Indica una classe, nella quale i suoi oggetti sono sequenze video che possono essere riprodotte, bloccate, ecc.
Window 	Classi	Indica una classe, nella quale i suoi oggetti hanno assegnata un'area dell'interfaccia utente, dove i frameset o gli oggetti di presentazione saranno visualizzati. Le finestre possono essere spostate, ridotte ad icona, ridimensionate.

2 Stereotipi per modellare caratteristiche adattative

Il profilo UML definisce i seguenti stereotipi per modellare le funzionalità adattative per sistemi multimediali adattativi. Questi sono usati nella costruzione del modello della struttura di navigazione, nel modello adattativo, e nel modello di presentazione.

STEREOTIPI/ICONE	APPLICATI A	SIGNIFICATO
Adapted language	Classi di presentazione	Indica che gli oggetti di presentazione testo saranno presentati in un linguaggio che dipende dal modello dell'utente.
annotated	Menu item Index item Anchor	Indica che un index item o un menu item o un'ancora hanno una notazione visuale particolare per indicare la loro importanza.
Direct guidance	Guided tour Anchor	Indica che il sistema deciderà quale sarà il miglior nodo d'arrivo.
Layout variant	Classi di presentazione	Indica che il layout dell'oggetto di presentazione dipende dalla scelta fatta dall'utente.
Passive navigation	Associazioni	Indica che la navigazione segue le associazioni eseguite dal sistema, per esempio quando l'utente rimane inattivo.
removed	Menu item Index item Anchor	Indica che il sistema decide che un item o un'ancora sono irrilevanti per un utente, di solito queste verranno rimosse.
Rule 	Classi	Indica una classe, nella quale i suoi oggetti contengono delle regole che determinano come aggiornare il modello dell'utente, come trovare dei contenuti appropriati e come adattate l'applicazione.
sorted	Menu item Index item Anchor	Indica che gli oggetti corrispondenti dipendono da un gruppo di item o d'ancore che sono ordinate secondo la loro importanza.
User behaviour 	Classi	Indica una classe, nella quale i suoi oggetti contengono il risultato di osservazioni date dall'utente.

CORRISPONDENZA FRA METACLASSI SPECIFICHE DI UWE E UML STANDARD

METACLASSE UWE	CORRISPONDE A	TAGGED VALUE	
		Nome	Valore
ConceptualClass	Classe	Uweclasstype	ConceptualClass
		Navigationclass	0 se non è una classe di navigazione; altrimenti UUID della classe di navigazione
NavigationClass	Classe	Uweclasstype	Navigationclass
		Conceptualclass	0 se non è una classe concettuale; altrimenti UUID della classe concettuale
		presentationclass	0 se non è una classe di presentazione; altrimenti UUID della classe di presentazione
PresentationClass	Classe	uweclasstype	presentationClass
		navigationclass	0 se non è una classe di navigazione; altrimenti UUID della classe di navigazione
AccessPrimitive	Classe	accessprimitivetype	Tipo della primitiva d'accesso
		presentationclass	0 se non è una classe di presentazione; altrimenti UUID della classe di presentazione

METACLASSE UWE	CORRISPONDE A	TAGGED VALUE	
		Nome	Valore
MenuItem	Classe	ismenuitem	true
		caption	Titolo del menu item
		navigationelement	0 se non è un elemento di navigazione; altrimenti UUID dell'elemento di navigazione
		menu	0 se non è un menu; altrimenti UUID del menu

BIBLIOGRAFIA

- [1] ArgoUML. <http://www.tigris.org>.
- [2] ArgoUWE. <http://www.pst.informatik.uni.muenchen.de/projekte/argouwe>.
- [3] Balestri R. *IP Exchange Procedures* – ADCS 73651.
- [4] Baltau M. *Directory Services*. Politecnico di Torino, Dipartimento Automatica e Informatica.
- [5] Baumeister H., Koch N., Mandel L. *Towards a UML Extension for Hypermedia Design*. Proceedings of The Unified Modeling Language Conference: Beyond the Standard (UML'99). France R. and Rumpe B. (Eds). LNCS 1723, Springer Verlag, 1999, 614-629.
- [6] Berner S., Glinz M., Joos S. *A Classification of Stereotypes for Object-oriented Modeling Languages*. In Proceedings UML'99–The Unified Modeling Language: Beyond the standard Conference. France R. and Rumpe B.(Eds.).LNCS 1723. Springer Verlag, 1999, 249-264.
- [7] Booch G. *Object-oriented Analysis and Design with Applications*. Cummings Publishing Company, 1994.
- [8] Booch G., Rumbaugh J., Jacobson I. *The Unified Modeling Language: A User Guide*. Addison Wesley, 1999.
- [9] Connallen J. *Building Web Applications with UML*, Addison Wesley, 1999.
- [10]Choi W., Kent A., Lea C. Presad G., Ullman C. *PHP4 Guida per lo sviluppatore*. Hoepli Informatica, 2002.
- [11]Dei Rossi G. *LightWeight Directory Access Protocol (LDAP)*. Università di Venezia.
- [12]DivWeb. <http://divWeb-cmg.st.com>

- [13]EnterpriseDirectory.
http://coat.sgp.st.com/services/Enterprise_directory/index.html.
- [14]E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley, 2000.
- [15]Hennicker R., Koch N. *A UML-based Methodology for Hypermedia Design*. Proceedings of the Unified Modeling Language Conference, UML '2000, Evans A. and Kent S. (Eds.). LNCS 1939, Springer Verlag, 2000, 410-424.
- [16]Hennicker R., Koch N. *Systematic Design of Web Applications*. Unified Modeling Language: System Analysis, Design, and Development Issues", Siau K. & Halpin T. (Eds.). Idea-Group Publishing, 2001, 1-20.
- [17]Hennicker R., Koch N. *Modeling the User Interface of Web Applications with UML*. In Practical UML-Based Rigorous Development Methods, Workshop of the pUML-Group at the UML '01, Gesellschaft für Informatik, Köllen Druck-Verlag, 2001.
- [18]Jacobson I., Booch G., Rumbaugh J. *The Unified Software Development Process*. Addison Wesley, 1999.
- [19]Knapp A., Koch N., Moser F., Zhang G. *ArgoUWE: A CASE Tool for Web Applications*. Ludwig-Maximilians-University Munich, Institute of Computer Science, EMSISE'03, 2003.
- [20]Koch N. *Hypermedia Systems Development based on the Unified Process*. Ludwig-Maximilians-University Munich, Institute of Computer Science, Technical Report 0003, 2000.
- [21] Koch N. *Software Engineering for Adaptive Hypermedia System: Reference Model, Modeling Techniques and Development Process*. PhD-Thesis, Ludwig-Maximilians-Universität München, Verlag UNI-DRUCK 200, 2001.

- [22] Koch N., Baumeister H., Hennicker R., Mandel L. *Extending UML to Model Navigation and Presentation in Web Applications*. Workshop on the UML and Modeling Web Applications, UML'2000, 2000.
- [23] Koch N., Helmerich A. *Information Services Procurement for WebEngineering* Ten Hagen & Stam Verlag, 2000.
- [24] Koch N., Mandel L. *Using UML to Design Hypermedia Applications*. Ludwig-Maximilians-University Munich, Institute of Computer Science, Technical Report 9901, 1999.
- [25] KOCH N., KRAUS A., HENNICKER R. *The Authoring Process of the UML-based Web Engineering Approach*, In First International Workshop on Web-Oriented Software Technology IWWOST'2001, Valencia, to appear, 2001.
- [26] KOCH N., Kraus A.. *The Expressive power of UML-based Web Engineering* In: Daniel Schwabe, Oscar Pastor, Gustavo Rossi, Luis Olsina (eds.), Proc. 2nd Int. Wsh. Web-Oriented Software Technology (IWOOST'02), CYTED, 2002.
- [27] Koch N., Kraus A.. *Towards a Common Metamodel for Web Applications*. In: Proc. 3rd Int. Conf. Web Engineering (ICWE'03).Lect. Notes Comp. Sci. Springer. To appear, 2003.
- [28] KRAUS A., KOCH N. *Generation of Web Applications from UML Models using an XML Publishing Framework*, to be published in the Conference Proceeding of the Integrated Design and Process Technology Conference, IDPT'2002, Pasadena, 2001.
- [29] Kruchten P. *The Rational Unified Process: An Introduction*. Addison Wesley, 1998.
- [30] LIEBERMAN B. *UML Activity Diagrams: Versatile Roadmaps for Understanding System Behavior*, Rational Edge Electronic Magazine for the Rational Community, 2001.
- [31] Malakowski M. *Mysql guida completa*. Apogeo, 2001.

- [32]MARKOPOULUS P. *Supporting Interaction Design with UML, Task Modelling* In Proceedings of the TUPIS'2000 Workshop at the UML'2000, 2000.
- [33]MARKOPOULUS P. *Modelling User Tasks with the Unified Modelling Language*, to appear, 2002.
- [34]NUNES J. N., CUNHA J. F. *Towards a UML Profile for Interaction Design: The Wisdom approach*, In Proceedings of the Unified Modeling Language Conference, UML'2000, Evans A. and Kent S., Eds., LNCS 1939, Springer Verlag, 2000, 100-116.
- [35]PATERNÒ F. *ConcurTaskTrees and UML: how to marry them?*, In Proceedings of the TUPIS'2000 Workshop at the UML'2000, 2000.
- [36]Pinheiro da Silva P., Paton N.: *UMLi: The Unified Modeling Language for Interactive Applications*. In Proceedings «UML» 2000, Evans, A., Kent, S. (Eds), LNCS, Vol. 1939. Springer-Verlag, 2000, 117-132.
- [37]Rational Software Corporation. *Rational Unified Process*
<http://www.rational.com/products/rup/index.jsp>.
- [38]Rational Software et al. *UML Notation Guide, version 1.1*.
<http://www.omg.org>, Sep 1, 1997. OMG Document ad/97-08-05.
- [39]Sano D. *Designing Large-Scale Web Sites: A Visual Design Methodology*. Wiley Computer Publishing, 1996.
- [40]Sauer S. & Engels G. *Extending UML for Modeling Multimedia applications*. Proceedings of the IEEE Symposium of Visual Languages – VL'99, IEEE Computer Society, 1999, 80-87.
- [41]Schneider G., Winters J. *Applying Use Cases: A Practical Guide* Addison-Wesley, Object Technology Series, 1998.

- [42]Sun Microsystems. Java Foundation Classes.
<http://java.sun.com/products/jfc/>.
- [43]The Group Services. <http://edirectory.st.com/groups>.
- [44]UML Version 1.3 *Unified Modeling Language*. The Object Management Group. 1999.
<http://www.omg.org>.
- [45]UML Version 1.4. *Unified Modeling Language*. The Object Management Group (OMG), 2001. <http://www.omg.org>
- [46]VAN HARMELEN M. *Interactive System Design Using Oo&hci Methods*, In Object Modeling and User Interface Design, van Harmelen M., Ed., Addison Wesley, 2001, 365-427.
- [47]Zhang G. *Case Support For Modelling Web Application*. Diploma thesis, Ludwig-Maximilians-Universität München, 2002.
<http://www.pst.informatik.unimuenchen.de/projekte/argouwe/argouwe-0.8.1a.pdf>.

