

Stato del Documento	Rif.	
Draft	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 1 di 80

# WEB@WORK - White Paper

## 1998 KME, 2000 ArchDoc, 2001 WEB@WORK

“an information odyssey”

### Revisioni del documento

Versione	Data	Descrizione	Autore
1.0	04/05/2001	Emissione Prima Bozza	Marco Caressa
2.0	13/06/2001	Draft	Marco Caressa <sup>1</sup> , Andrea Manieri <sup>2</sup> (§ 4.3, 4.4)

Inviare commenti e contributi a:

[web-at-work@eng.it](mailto:web-at-work@eng.it)

<sup>1</sup> E-business Solution Architect presso il Laboratorio di Roma di Engiweb.com S.p.A.

<sup>2</sup> Ricercatore presso il Laboratorio di Ricerca e Sviluppo di Roma di Engineering Ingegneria Informatica.

<b>Stato del Documento</b>	<b>Rif.</b>	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 2 di 80

1	Premessa.....	4
2	Il contesto tecnologico/organizzativo .....	5
2.1	Delocalizzazione dei processi .....	5
2.2	Domini di interazione, knowledge workers e processi di workflow .....	8
2.3	Gestione della conoscenza .....	10
3	L'eredità del KME .....	12
4	L'evoluzione della specie.....	14
4.1	L'occasione Archdoc .....	14
4.2	Obiettivi di Engiweb .....	16
4.3	Obiettivi del Laboratorio di Ricerca e Sviluppo di Engineering .....	17
4.3.1	I progetti in corso.....	17
4.4	Sinergie .....	19
4.4.1	I punti di contatto.....	19
4.4.2	I contributi del laboratorio.....	19
4.5	Evoluzione della specie: da Archdoc a WEB@WORK .....	22
5	L'architettura di WEB@WORK.....	24
5.1	Panoramica .....	24
5.2	Architettura logica.....	24
5.3	Architettura applicativa e architettura operativa .....	25
5.3.1	Encoders/Decoders e Interaction Manager .....	28
5.3.2	Contents Manager.....	28
5.3.3	Adapters .....	29
5.3.4	Security Services e Profile Manager .....	30
6	WEB@WORK: meta-modello e interfacce funzionali.....	32
6.1	Un modello ad oggetti.....	32
6.2	La piattaforma tecnologica e il modello dei dati .....	33
6.3	La piattaforma tecnologica e l'ambiente di sviluppo delle API .....	33
7	Il "Core Engine".....	35
7.1	I Metadati di supporto al modello ad oggetti .....	35
7.1.1	Analisi dei requisiti.....	35
7.1.2	Disegno del sistema.....	37
7.2	Il sistema di autorizzazioni .....	45
7.2.1	Visione generale.....	45
7.2.2	Analisi dei requisiti.....	46
7.3	Utenti e gruppi .....	47
7.3.1	Visione generale.....	47
7.3.2	Analisi dei requisiti.....	48
7.3.3	Disegno del sistema.....	50
7.4	Gestione della sicurezza .....	54
7.4.1	Visione generale.....	54
7.4.2	Disegno del sistema.....	55
7.4.3	API.....	58
8	I Moduli .....	60

<b>Stato del Documento</b>	<b>Rif.</b>	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 3 di 80

8.1	Modulo "Workflow" .....	60
8.1.1	Visione generale.....	60
8.1.2	Analisi concettuale .....	61
8.1.3	Il modello dei dati .....	63
8.1.4	Interfacce (cenni).....	63
8.2	Modulo "Information Warehouse" .....	66
8.2.1	Visione generale.....	66
8.2.2	Produzione di contenuti per il Web .....	67
8.2.3	Il modello ad oggetti .....	69
8.2.4	Organizzazione dei contenuti.....	71
8.2.5	Relazioni tra contenuti.....	72
8.2.6	Presentazione dei contenuti.....	73
8.2.7	Pubblicazione dei contenuti .....	73
8.2.8	Modello dei dati e interfacce .....	74
8.3	Modulo "Messaging" .....	74
8.3.1	Strutture di dati e interfacce .....	75
9	La realizzazione del sistema .....	76
9.1	Stime di massima .....	78
10	Appendice .....	79

# 1 Premessa

Si assiste oggi ad una generale convergenza verso le soluzioni *web-based* delle architetture riguardanti le piattaforme applicative. Oltre a rappresentare la linea evolutiva imboccata decisamente dal mercato informatico, va evidenziato come tali architetture consentano una fruibilità dell'informazione ed una omogeneità d'impiego grandemente superiori rispetto alle applicazioni di un passato anche recente.

Quando si dispone di un certo numero di applicativi *web-based* (o di accessi *web-based* ad applicativi tradizionali) è naturale pensare ad una loro organizzazione ed omogenizzazione all'interno di una portale di servizio.

Si ha così la possibilità di dotare gli operatori dell'azienda e gli utenti esterni di un unico punto di accesso a tutte le applicazioni che via via si adeguano o si aggiungono a tale architettura, a patto, naturalmente, che questa presenti dal punto di vista funzionale caratteristiche di modularità intrinseca. Ciò consente di erogare a tutti gli operatori interessati gli opportuni servizi informativi, tecnico-amministrativi e di lavoro collaborativo in modo efficace, pervasivo, ricco e flessibile. Tali servizi possono essere in prima istanza classificati nelle seguenti categorie:

- Strumentazione (motori di ricerca, alberi di decisione, etc.) attraverso la quale fruire delle informazioni sia statiche che dinamiche (documenti, contenuti di database, etc.) organizzate in una infrastruttura generale a supporto della rappresentazione della "*conoscenza*" aziendale (tassonomie, mappe concettuali, etc.).
- Servizi atti a definire ed individuare un *ambiente di lavoro collaborativo* per tutti gli operatori che condividano un dominio di interazioni strutturate e non strutturate che coinvolgono un insieme di soggetti, a prescindere dalla loro partecipazione ai processi di lavoro.

Questo documento si propone due obiettivi. Da un lato delineare i mutamenti dello scenario tecnologico/organizzativo delle aziende. Dall'altro, sulla base delle esperienze maturate negli ultimi due anni prima dal Laboratorio di Ricerca e Sviluppo di Engineering e successivamente da Engiweb.com, proporre una infrastruttura di servizi applicativi generali a supporto di soluzioni *web-based* che rappresenti un'evoluzione dei prodotti e delle soluzioni sin qui sviluppate.

In tale disamina vanno inizialmente evidenziati due aspetti ritenuti particolarmente rilevanti:

- la delocalizzazione dei processi di lavoro e la definizione dei *domini di interazione*
- la gestione della *conoscenza* all'interno di un'organizzazione.

## 2 Il contesto tecnologico/organizzativo

### 2.1 Delocalizzazione dei processi

Oggi l'esplosione delle tecnologie *ICT*, ed in particolare dell'*e-business*, obbliga ad un profondo ripensamento degli approcci metodologici allo sviluppo delle tecnologie organizzative nel contesto d'uso dei sistemi di cooperazione. L'ambiente di lavoro si identifica sempre meno con la sede fisica di un'organizzazione e sempre più con la sua *rete*, e sempre più spesso questa è una *intranet* interconnessa alla rete *internet*.

E' in corso una delocalizzazione strutturale dei processi di lavoro. Il *luogo* dove i processi risiedono si approssima sempre più ad uno spazio costituito dal substrato della rete aziendale (*intranet*) e in quello delle reti esterne (*extranet*, *internet*). Nella generalità delle organizzazioni questo è ovviamente un processo strutturale di tendenza, che spesso le abitudini culturali ed organizzative tendono ad occultare. Tuttavia, questa mutazione strutturale non può essere trascurata, poiché impatta profondamente sul concetto stesso di organizzazione tradizionale.

Nel termine *rete* si incrociano tecnologie organizzative, informatiche e telematiche che ridisegnano complessivamente l'organizzazione, che passa dal tradizionale assetto gerarchico ad uno basato sui processi e sulla cooperazione produttiva.

Il fenomeno può essere compreso osservando l'evolversi dei domini in cui si svolgono i processi di produzione in un'organizzazione. I domini coinvolti sono:

- Dominio dei processi materiali

I processi materiali possono essere definiti come attività essenziali rivolte al mondo fisico. Questi implicano spostamento fisico di oggetti e cambiamenti di stato. In una tradizionale impresa manifatturiera questo si riassume nella trasformazione ed assemblaggio di oggetti fisici in un prodotto. Nella tradizione industriale questo è considerato il dominio primario, dal punto di vista metodologico il richiamo è ai contributi di Taylor e Ford all'organizzazione industriale.

- Dominio dei processi informativi

I processi materiali, tuttavia, non coprono tutti gli aspetti essenziali delle attività di business<sup>3</sup> di un'organizzazione. Negli ultimi 50 anni è cresciuta esponenzialmente l'importanza dei processi informativi che, con la crescita delle tecnologie informatiche, sono diventati l'attività più rilevante nelle organizzazioni. Gli stessi processi materiali possono essere rappresentati come processi informativi utilizzando simboli ed astrazioni appropriate. I processi informativi hanno perciò assorbito, di fatto, quelli materiali e in numerose attività di business sono oramai indistinguibili. La gran parte delle tecnologie informatiche oggi disponibili sono orientate proprio a questo dominio.

- Dominio dei processi di business

L'informazione ha significato, per un'organizzazione, solo se qualcuno può fare qualcosa con essa, quindi assume valenza se associata, in senso lato, ad un'attività di business.

La dimensione che il dominio dei processi di business aggiunge è quella degli atti linguistici e delle conversazioni<sup>4</sup> che questi generano ed è quella che dà senso all'informazione contenuta nei database e nei documenti e più in generale alle attività che si svolgono nell'organizzazione. Il complesso delle tecnologie *ICT*, consentendo di svincolare le conversazioni dalla contemporaneità e dalla compresenza fisica, slega l'organizzazione dal luogo fisico spostando le conversazioni nello spazio della rete.

<sup>3</sup> Con in termine *business* si denotano le attività effettuate da un'organizzazione per la sua sopravvivenza ed autonomia. Vedi: Raul Medina-More – Gita J. Melkote - Fernando Flores, "Business Process: Design and Development", in Y. Jayachandra, *Re-Engineering the networked Enterprise*, McGraw-Hill Inc, 1993

<sup>4</sup> Con il termine conversazioni si intendono qui le "reti di cooperazione" presenti in un'organizzazione.

Lo sviluppo delle tecnologie informatiche ha fatto quindi emergere nelle organizzazioni domini un tempo poco conosciuti, o comunque considerati secondari, che implicano una riconsiderazione dei meccanismi di produzione di valore e di servizi.

Infatti è evidente che il dominio dei processi di business è primario rispetto al dominio dei processi informativi e a quello dei processi materiali, dato che *è in esso che viene effettivamente orientata e realizzata l'organizzazione*, definendo ciò che in essa viene percepito e fatto. In effetti la realizzazione dei processi di business è il “luogo” stesso della realizzazione dell'organizzazione e della sua autonomia.

In questo contesto è altrettanto evidente che l'uso intensivo delle tecnologie *ICT* abilita strutturalmente nuove possibilità organizzative. La sostanziale indipendenza dei processi di lavoro dalla compresenza e contemporaneità fisica delle persone implica una ristrutturazione profonda che modifica il modo stesso in cui viene definita e percepita l'organizzazione, i suoi confini e i suoi processi vitali.

Questo processo in atto rende necessario considerare l'organizzazione in una prospettiva differente: non essendo più solo *la sede* il luogo fisico dove si svolgono i processi di lavoro vitali ma la sua rete, l'aspetto strutturale parte dalla *delocalizzazione* dei processi piuttosto che dalla *decentralizzazione* dei luoghi di lavoro.

Un processo può essere definito come: “*un gruppo strutturato e misurato di attività progettate per produrre un risultato specifico per un particolare cliente o mercato*”<sup>5</sup>.

Un processo di lavoro delocalizzato è un processo che si svolge sulla rete o sulle reti interconnesse utilizzate dall'azienda. La delocalizzazione presuppone quindi un'architettura connettiva (la rete) a cui tutti i membri dell'organizzazione (ma sempre più spesso anche tra diverse organizzazioni e tra organizzazioni e mercati/clienti) hanno accesso indipendentemente dal luogo fisico di lavoro. Dal punto di vista organizzativo la rete è quindi lo spazio topologico dove si svolgono le interazioni tra membri dell'organizzazione.

		TEMPO	
		=	≠
LUOGO	=	1 Stesso Luogo Modo Sincrono	2 Stesso Luogo Modo Asincrono
	≠	3 Luoghi Diversi Modo sincrono	4 Luoghi Diversi Modo Asincrono

#### La matrice tempi-luoghi

La comunicazione in una rete avviene attraverso una serie di strumenti che consentono di trasferire le interazioni generate dai membri in modo sia sincrono che asincrono comprendendo la classica matrice tempi/luoghi. La copertura completa di questa matrice è un risultato delle tecnologie *ICT* ed è uno degli aspetti che ha sconvolto i modelli comunicativi delle organizzazioni tradizionali.

Una delle conseguenze della copertura della matrice è infatti la riduzione del costo del coordinamento (aumenta la facilità della comunicazione e se ne riduce il peso). Questa è una delle cause del crollo dell'efficacia relativa delle tradizionali organizzazioni gerarchico-burocratiche.

<sup>5</sup> Thomas H. Davenport, *Process Innovation: Reengineering Work through Information Technology*, Harvard Business School Press, 1992.

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 7 di 80

La facilità e la completezza delle situazioni comunicative coperte implica la possibilità di aumentare la connettività interna delle organizzazioni senza renderle instabili<sup>6</sup> (a causa dell'eccessivo tempo ed energia necessaria al coordinamento).

La gerarchia è una strategia che riduce la varietà nelle organizzazioni, dandogli unità e coerenza, a prezzo del loro irrigidimento. Una volta che sia possibile aumentare la connettività delle organizzazioni a costi bassi la gerarchia diventa un ostacolo alla reattività ed all'efficacia delle stesse.

Dal punto di vista della modalità con cui sono espresse le comunicazioni in una organizzazione e della tipologia del contesto in cui sono espresse, possiamo considerare la seguente tassonomia:

- Per la **modalità** di erogazione:

*scritta*

*vocale*

*visiva*

- Per il **contesto** in cui la comunicazione è espressa:

*diretto*

Prodotto da persona a persona all'interno di una *conversazione* propriamente detta. Il parlante e l'ascoltatore hanno un rapporto diretto e l'interazione prende la forma dell'*atto linguistico* (*richiesta, dichiarazione, promessa, ecc.*) .

*mediato*

Le comunicazioni sono dirette ad utenti generici non individuati immediatamente, in questo caso non vi è una conversazione in atto in senso proprio, ma solo la pubblicazione di notizie e dichiarazioni che l'ascoltatore fruirà per i propri scopi relativamente indipendenti. Es. consultazione delle informazioni in database o pubblicazione di documenti, saggi, analisi, previsioni, conferenze o presentazioni ad un pubblico.

La delocalizzazione di processi non rappresenta però una semplice aggiunta di strumenti comunicativi. Essa, come è stato detto, costituisce uno spazio di interazione tra persone diverso da quello tradizionale che richiede la creazione di adeguate forme organizzative e di un'adeguata integrazione degli strumenti disponibili.

È infatti evidente come la virtualizzazione della comunicazione renda inservibile quell'insieme di strumenti organizzativi basati sulla gerarchia che caratterizzano la tradizione. L'implementazione di processi di lavoro delocalizzati richiede quindi un'adeguata revisione dei principi organizzativi e l'applicazione di **modelli basati sui processi**.

Tuttavia il semplice passaggio ad organizzazioni basate sui processi non è sufficiente, in quanto il processo di lavoro rischia di essere ridotto al tentativo di applicazione di una procedura standardizzata. Vi è quindi una ulteriore dimensione organizzativa che le tecnologie di rete enfatizzano: quella **dell'autonomia e dell'impegno**. In altre parole:

Viene accordata una forte autonomia nell'esecuzione (conta essenzialmente l'obiettivo ed il tempo concordato)

Il controllo delle attività viene effettuato in base all'esecuzione degli impegni ed alla soddisfazione e non in base al controllo gerarchico ed al semplice rispetto di procedure.

Ciò consente di introdurre il concetto di *domini di interazione*.

---

<sup>6</sup> Si veda il concetto di "equilibrio plastico" in §6.2.5 di *PESI\_exec\_summ.doc*

## 2.2 Domini di interazione, knowledge workers e processi di workflow

L'implementazione dei processi di lavoro delocalizzati implica non solo la definizione "istitutiva" dei processi (come sequenza di attività) ma anche la definizione del particolare dominio di interazione dove i processi si svolgono.

In sintesi non è sufficiente, nella progettazione dei processi di business, limitarsi all'identificazione di ruoli e sequenze di attività, di quello cioè che comunemente viene definito *workflow*. Il passo preliminare è riconoscere il campo in cui si realizzano le interazioni che determinano l'organizzazione. Questo è vitale quando gli aspetti fisici (ufficio, sede aziendale) perdono la capacità di delimitare lo spazio in cui si svolgono i processi di lavoro.

Per quanto detto nei paragrafi precedenti, l'implementazione di processi di lavoro delocalizzato implica il passaggio da forme organizzative gerarchico-burocratiche a forme centrate sui processi e sull'autonomia degli operatori. E' oggi evidente come il patrimonio di *conoscenza (knowledge)* sia diventato il più importante fattore di sviluppo economico delle organizzazioni, e come tale patrimonio risieda primariamente nelle *menti* degli operatori (*knowledge workers*).

Nell'organizzazione tradizionale il lavoratore è sostanzialmente visto come una macchina che esegue istruzioni precodificate. Il lavoro è in questo contesto di tipo *esecutivo*, con una bassa autonomia: in caso di problemi nello svolgimento del lavoro la decisione (se non rientra nei casi precodificati) viene spostata al livello gerarchico superiore. L'abitudine culturale ha fatto sì che anche lavori *non esecutivi* siano stati immersi in organizzazioni disegnate su base sostanzialmente gerarchico-burocratica. Inoltre, paradossalmente, la crescente automazione ha ridotto enormemente la stessa necessità di organizzazioni dove il lavoratore sia trattato come una macchina.

La delocalizzazione di processi comporta invece un incremento di autonomia da parte del *knowledge worker* che prende decisioni, potendo consultarsi con rapidità con la sua rete di cooperazione (team, gruppi di lavoro, clienti, fornitori) senza coinvolgere una escalation gerarchica.

L'immediatezza della comunicazione insieme alla disponibilità di una informazione molto completa sullo stato del processo è il fattore rilevante che aumenta la capacità di risoluzione di problemi con una bassa utilità di strutture gerarchiche.

Le caratteristiche generali di un ambiente di lavoro per la delocalizzazione dei processi possono essere sintetizzate in:

- L'ambiente deve collegare in un unico spazio tutti i *knowledge worker* coinvolti nel dominio di interazione specifico.
- L'ambiente deve fornire strumenti di comunicazione sia sincroni (come *chat*, *whiteboard*, *audio/videocomunicazione* ecc.) che asincroni (*posta*, *newsgroup*, *messaggi broadcast*).
- L'ambiente deve fornire la possibilità di condividere i documenti interrelati al dominio gestendo funzioni di *ownership* per i documenti in lavorazione, fino alla gestione del versionamento, a seconda delle esigenze.
- L'uscita dall'ambiente deve coincidere con l'uscita dal dominio di interazione.
- L'ambiente deve fornire strumenti per la comunicazione esterna al dominio. Deve cioè fornire la possibilità di interagire con i "clienti" dei prodotti realizzati nel dominio.

Un ambiente *Internet/Intranet* risponde a questi requisiti di base, permettendo di accedere a tutte le comunicazioni relative ad un dominio generando così un *collaboratory*.

Secondo la definizione di William Wulf (1989) un *collaboratory* è un "*centro senza pareti*" in cui gli utenti "*conducono la loro ricerca senza legami geografici, interagendo con i colleghi, avendo*



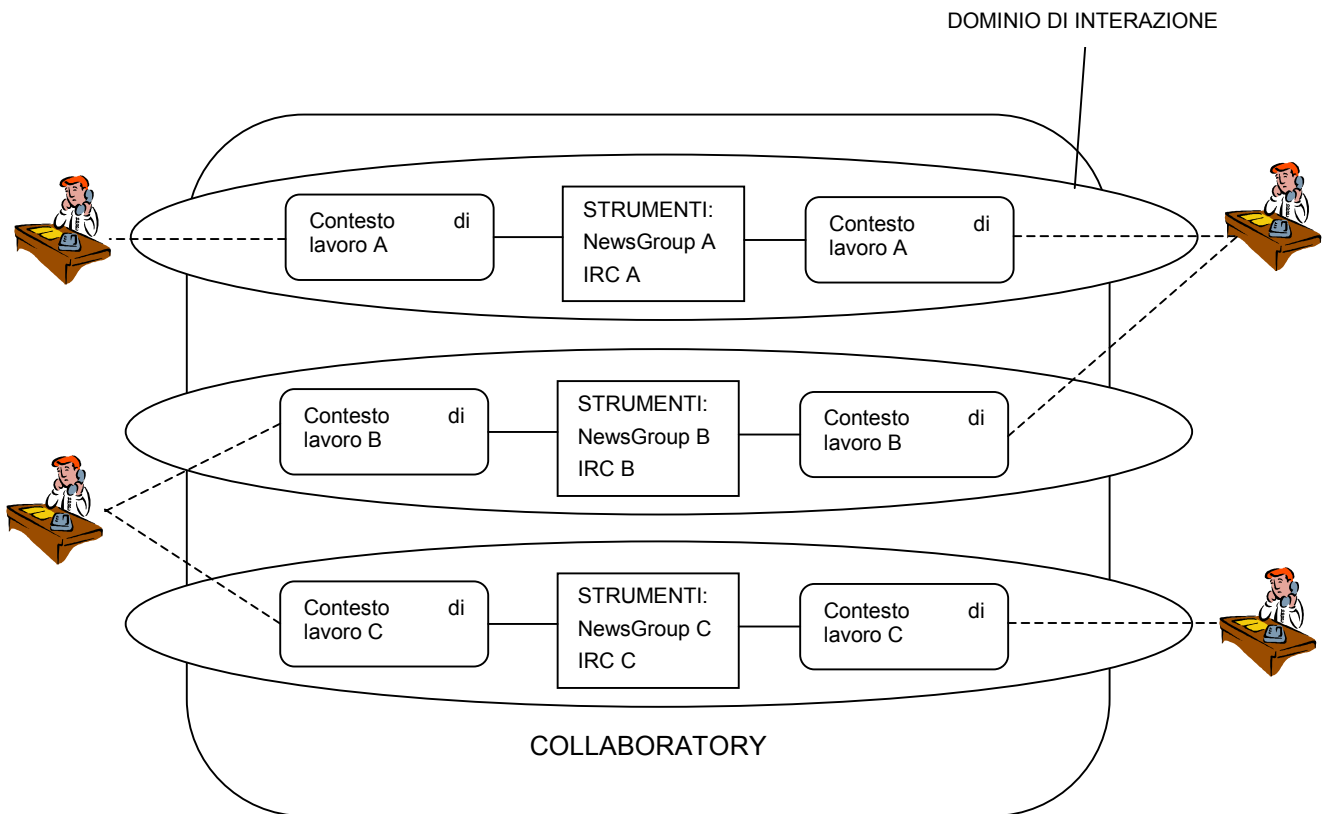
accesso alla strumentazione, mettendo in comune i dati e le risorse di calcolo ed utilizzando le informazioni delle biblioteche digitali<sup>7</sup>.

Ambienti di questo tipo consentono di coprire la varietà di comunicazioni non strutturate necessarie alla realizzazione della delocalizzazione dei processi di lavoro. La loro implementazione organizzativa parte da un adeguato disegno dei domini di interazione.

Per la delimitazione del dominio il primo passo è l'individuazione dell'area di interesse generata dall'orientamento reciproco che emerge dalle conversazioni (esempio: realizzazione di un certo prodotto, fornitura di un determinato servizio, ecc.).

Il secondo passo consiste nell'individuazione dei partecipanti al dominio di interazione e può essere sintetizzata nei seguenti punti<sup>8</sup>:

- Va considerato parte del dominio chiunque dentro l'organizzazione produce o tratta informazioni riguardanti l'ambito d'interesse.
- Va considerato parte del dominio chi partecipa a produzioni che riguardano quell'ambito.
- Vanno inserite nel dominio tutte le fonti informative disponibili riguardanti l'ambito.
- Non è parte del dominio chi osserva quell'ambito come livello gerarchico superiore (l'osservatore genera un diverso dominio di interazione).
- Non è parte del dominio il cliente finale dell'organizzazione.



<sup>7</sup> Vedi: Nancy Ross-Flanigan, "Collaborare in rete" su *Technology Review* (edizione italiana dell'omonima rivista del M.I.T.) Anno XI n°1 Maggio-Giugno, 1998

<sup>8</sup> La definizione del dominio di interazione va ovviamente riverificata una volta che i processi di lavoro siano analizzati. I confini dei processi di lavoro e quelli dei domini di interazione non possono essere contraddittori.

Un'organizzazione sarà formata da diversi domini di interazione a seconda degli ambiti di cui si occupa e delle funzioni che in essa vengono assolate. Un *knowledge worker* partecipa normalmente a diversi domini di interazione internamente all'organizzazione.

Tipicamente vi saranno domini di interazione su attività continuative (ad esempio sulle problematiche di assistenza su un prodotto/servizio) o domini generati da progetti specifici o da tipologie di progetti. A questi vanno aggiunti domini sulle questioni amministrative dell'organizzazione.

La definizione dei domini di interazione permette di comprendere come disegnare gli ambienti di cooperazione. Si noti che a questo livello si è al di sopra dei processi di lavoro in quanto il dominio raccoglie l'intero campo di comunicazione, compresa quella non strutturata, dell'organizzazione. I domini di interazione hanno inoltre un importante significato rispetto al *knowledge management* dell'organizzazione, in quanto rappresentano implicitamente le aree di conoscenza effettiva dell'organizzazione. In questo senso costituiscono un importante legame operativo tra la conoscenza creata dall'organizzazione e i processi di lavoro di questa consentendo di disegnare un legame operativo tra i due aspetti.

Il disegno dei domini di interazione risponde alla domanda su chi, in un organizzazione, crea ed utilizza la conoscenza e quindi dove questa va catturata ed a chi va fornita.

Il passo successivo, una volta determinati i domini di interazione (che guidano la progettazione e realizzazione degli spazi elettronici o collaborativi), è disegnare i processi di lavoro (*workflow*) che attraversano questi spazi e quindi definire la struttura ricorrente delle conversazioni che si svolgono nei domini.

I domini di interazione definiscono lo spazio in cui si realizzano i processi di *workflow* ma non definiscono i processi stessi. Definiscono, cioè, lo spazio dei *collaboratory* (spazi per la comunicazione *non strutturata*). Vi sono però, all'interno di questi, strutture ricorrenti che definiscono i processi di *workflow* e quindi danno luogo a conversazioni strutturate nei domini.

Queste strutture emergono da regolarità nell'uso del linguaggio derivate dall'accoppiamento reciproco tra coloro che lo utilizzano negli appropriati domini di interazione.

E' possibile in questa logica tracciare il corso di una conversazione. Questa può comprendere al suo interno messaggi di posta elettronica, fax, immagini grafiche, query, documenti. Essa ha però una struttura fondamentale stabile, sulla quale è possibile modellare l'uso degli strumenti, costituita da un insieme di *stati* e *transizioni*.

## 2.3 Gestione della conoscenza

L'obiettivo fondamentale è dotare i *knowledge workers* della possibilità di condividere efficacemente, attraverso i domini di interazione, sia la propria capacità intellettuale globale che la conoscenza in loro possesso.

Le componenti principali che consentono la gestione della conoscenza aziendale sono state introdotte e delineate nel precedente paragrafo. I processi di lavoro sono delocalizzati e trovano attuazione nell'ambiente collaborativo (*collaboratory*) costituito dall'infrastruttura *Internet/Intranet*, il modello organizzativo passa da una struttura gerarchico-burocratica (albero) ad una fortemente interconnessa (grafo), sono enfatizzati gli aspetti relativi all'autonomia e all'impegno dei *knowledge workers*, che operano all'interno di domini di interazione ben definiti. Questi individuano lo spazio di interazione all'interno del quale trovano posto due tipologie di *conversazioni*:

- conversazioni informali e destrutturate realizzate con strumenti:
  - *sincroni*: chat, whiteboard, comunicazioni audio/video, ecc.
  - *asincroni*: posta, newsgroup, messaggi broadcast, forum, e-circle, ecc.
- conversazioni ricorrenti e strutturate: processi di *workflow*.

Altre due componenti essenziali del sistema di gestione della conoscenza sono costituite da:

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 11 di 80

- un "luogo" nel quale depositare i "pezzi di conoscenza" (*knowledge content*) che quotidianamente l'organizzazione produce (*knowledge repository*) mediante opportune forme di rappresentazione (ontologie, mappe concettuali, etc.)
- una strumentazione volta al *recupero* della conoscenza presente nei repository e nei database (es. motori di ricerca, sistemi di tassonomia, etc.) che tragga profitto dalle forme di rappresentazione appena citate.

Come appare ormai evidente, *la gestione della conoscenza* non si concretizza solo nell'applicazione di una tecnologia, ma implica profondi cambiamenti culturali nel modo con cui i membri dell'organizzazione percepiscono la conoscenza che sviluppano.

Una delle chiavi di successo di un'iniziativa di *collaboratory* è data dalla maturità dell'organizzazione nel considerare la conoscenza, e la possibilità di condividerla, come uno dei fattori critici della propria capacità complessiva di conseguire i propri obiettivi.

Nel caso di organizzazioni come le università l'importanza della gestione e della condivisione della conoscenza balza immediatamente all'evidenza. Esse sono i luoghi deputati alla produzione di conoscenza e pressoché tutto ciò che gestiscono è conoscenza.

La quantità di "*knowledge content*" presenti nelle organizzazioni è spesso ragguardevole. Senza pretesa di essere esaustivi si possono considerare :

- pubblicazioni
- monografie e periodici
- atti di convegni
- circolari
- normativa
- messaggistica
- documentazione varia
- altro

La possibilità di fruire in modo integrato ed organizzato di tutto questo insieme di conoscenza costituisce un sicuro valore aggiunto.

Il progetto complessivo di realizzazione di una applicazione Internet/Intranet che consenta di gestire la conoscenza aziendale non può prescindere dalle seguenti fasi:

- Censimento dei "blocchi di conoscenza" (*knowledge content*) presenti nell'organizzazione, identificazione del gestore istituzionale di ciascuno di essi (*knowledge owner*), della localizzazione (*knowledge site*) e del supporto su cui sono disponibili (*knowledge media*).
- Definizione della sequenza con cui rendere disponibili i "*content*".
- Identificazione dei "*content manager*", ossia delle persone che hanno il compito di garantire che l'informazione sia accurata, utile e di facile reperibilità e debbono curare il suo inserimento all'interno della mappa concettuale allo scopo di facilitarne il *recupero*.
- Identificazione di un campione di candidati fruitori di ciascun content e realizzazione delle interviste attraverso cui definire il contesto di fruizione (es. tassonomie, motori di ricerca, mappe concettuali).
- Attivazione del processo di "*content management*" attraverso cui avviene l'eventuale conversione dei documenti in formato digitale, l'aggiunta dei metadati che li renderanno fruibili all'interno del contesto di *knowledge management* ed il loro inserimento nel *database*.
- Definizione delle regole di manutenzione del *repository* (es. modalità di eliminazione dei *content* obsoleti).

### 3 L'eredità del KME

Nel capitolo precedente è stato evidenziato come lo sviluppo di sistemi per il trattamento di informazioni e di conoscenza con tecnologie *web-based* costituisca una delle maggiori possibilità di *business* per tutte le organizzazioni che si muovono nell'ambito dell'*Information-technology*.

Engineering recepì a suo tempo l'importanza di questi temi realizzando il sito intranet KME (*Knowledge Management of Engineering*), principalmente allo scopo di razionalizzare la raccolta e la gestione della conoscenza aziendale. Lo sviluppo di questo servizio, realizzato dal Laboratorio di Ricerca e Sviluppo ha segnato, pur tra i suoi alti e bassi, un risultato importante per tutti i dipendenti dell'azienda, per i quali il KME rappresenta oggi uno strumento di lavoro quotidiano.

I servizi di base che il sistema ha sempre fornito sono relativi alla possibilità di condividere con gli altri colleghi documenti tecnici di qualsiasi genere effettuandone l'*upload* verso un *repository* centrale, mettendo a disposizione un motore di ricerca *full-text* per l'individuazione di documenti potenzialmente utili, ed infine fornendo la possibilità di effettuare il *download* dei documenti giudicati utili.

I documenti possono essere classificati, all'atto della loro immissione, secondo una tassonomia dinamica (si possono aggiungere nuove categorie di classificazione, anche se non esiste attualmente uno strumento di amministrazione che consenta di farlo).

Nato come sistema per la gestione centralizzata di documentazione aziendale di varia natura (documenti tecnici, documentazione standard aziendale, manualistica di prodotti, etc.), il KME ha visto nel tempo l'aggiunta di nuove funzioni applicative (ricerche generalizzate, etc.) e di interi nuovi servizi (gestione centralizzata della contrattualistica di Engineering).

I nuovi sviluppi del sistema sono stati però effettuati sulla piattaforma tecnologica originaria, oramai obsoleta, poco aperta e di difficile gestione e manutenzione ma, soprattutto, basata su versioni attualmente non supportate di alcuni software di base utilizzati (Database Oracle 8.0.4 e Oracle Application Server 3.1.1.18).

Allo stato attuale, la gestione del KME presenta non pochi problemi tecnici legati a quanto appena detto. Tuttavia, le potenzialità dell'intuizione che a suo tempo si ebbe rimangono immutate. La gestione centralizzata e controllata della conoscenza aziendale è un problema che riguarda qualsiasi organizzazione di medie grandi dimensioni distribuita sul territorio. Generalizzando è possibile affermare che *qualunque* organizzazione sente oggi l'esigenza di gestire il proprio patrimonio di informazioni garantendone la condivisione controllata tra gruppi di utenti, laddove il concetto di "gruppo di utenti" va inteso nella sua accezione più generale (Es. la platea di Internet).

Poichè la piattaforma di riferimento di KME è quella Internet/Intranet, Engiweb.com ha raccolto il testimone dalla capogruppo circa la possibilità di proporre sul mercato soluzioni web per la gestione centralizzata delle informazioni e dei documenti basate su una evoluzione dei servizi forniti dal KME. Il riscontro è stato immediatamente positivo. E' infatti in fase di rilascio per il cliente SIC (Società Italiana Cauzioni) l'applicazione *Archdoc*, che rappresenta il primo tangibile risultato di evoluzione del sistema. In poco più di due mesi di sviluppo è stata realizzata un'applicazione web che mette a disposizione tutte le funzionalità di base del KME, estendone in modo significativo le possibilità applicative, come quella di poter dinamicamente definire la struttura dei documenti gestiti dal *repository* del sistema. Rispetto al KME il codice è stato completamente reingegnerizzato, passando dal linguaggio C al PL/SQL, razionalizzando alcuni servizi applicativi (*upload* e *download* di documenti) allo scopo di eliminare qualsiasi forma di manutenzione lato-client (utilizzo di *java servlet* al posto di componenti *active-x*).

Scopo di questo documento è delineare la possibilità di far evolvere Archdoc, estendendone le funzionalità, allo scopo di realizzare un framework che consenta la rapida ed efficace implementazione di servizi che consentano, tra gli altri, di:

<u>Stato del Documento</u>	<u>Rif.</u>	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 13 di 80

- *Definire in maniera dinamica la struttura di qualsiasi tipo di informazione (ivi compresi documenti) da trattare.*
- *Ottenere automaticamente le primitive (metodi) per operare su tali tipologie di informazione*
- *Definire dinamicamente le tassonomie di classificazione delle informazioni per trarne il dovuto profitto in fase di retrieval. Dovranno essere disponibili degli agenti software in grado di supportare o sostituire gli utenti nel processo di classificazione delle informazioni.*
- *Definire delle relazioni tra le tipologie di informazione che consentano di potenziare i meccanismi di recupero delle stesse (mappe concettuali).*
- *Produrre, validare e pubblicare contenuti informativi sul web (web-publishing).*
- *Gestire un sistema di autorizzazioni a livello di singolo utente del sistema, di singola operazione e di di singolo item (Es. se l'utente X abbia il diritto di effettuare l'operazione Y sull'item Z).*
- *Gestire processi di workflow sulle informazioni trattate nel sistema (Es. Gestione di protocolli o di pratiche, gestione dell'iter di un contratto, etc.).*
- *Gestire attività di collaborazione tramite il web (Es. Gestione di progetti software sia in termini di documentazione che di codice sorgente)*

I precedenti sono solo alcuni dei punti di interesse. Il quadro verrà completato in seguito, quando si descriveranno in dettaglio l'architettura del sistema e l'insieme completo dei servizi che questo sarà in grado di offrire.

## 4 L'evoluzione della specie

### 4.1 L'occasione Archdoc

Come accennato nella premessa, *Archdoc* rappresenta la prima evoluzione del sistema KME. Esso è stato sviluppato come personalizzazione del KME per la *Società Italiana Cauzioni*, e presenta una serie di caratteristiche che lo distinguono dal suo predecessore.

Si tratta di una soluzione per l'archiviazione e la gestione integrata della Documentazione Elettronica prodotta all'interno delle Unità Operative delle Aziende. Il sistema, estremamente versatile e personalizzabile, consente la gestione di archivi di qualsiasi tipo di documentazione (sia documenti elettronici – es. testi prodotti con programmi di videoscrittura standard – sia immagini di documenti cartacei acquisiti tramite scanner). L'obiettivo primario del sistema è quello di permettere la raccolta, classificazione e diffusione di tutte le informazioni prodotte all'interno dell'Azienda costituendo il nucleo centrale di una soluzione di *Knowledge Management*.

Grazie a questo sistema è possibile rispondere in modo efficace ad una serie di sfide che nell'era della globalizzazione acquistano una importanza sempre maggiore. Tra queste citiamo la capacità di poter migliorare ed accrescere giorno per giorno il proprio patrimonio di competenze e di conoscenza, un elemento che è ritenuto dagli esperti di organizzazione aziendale particolarmente rilevante per la sopravvivenza delle Aziende.

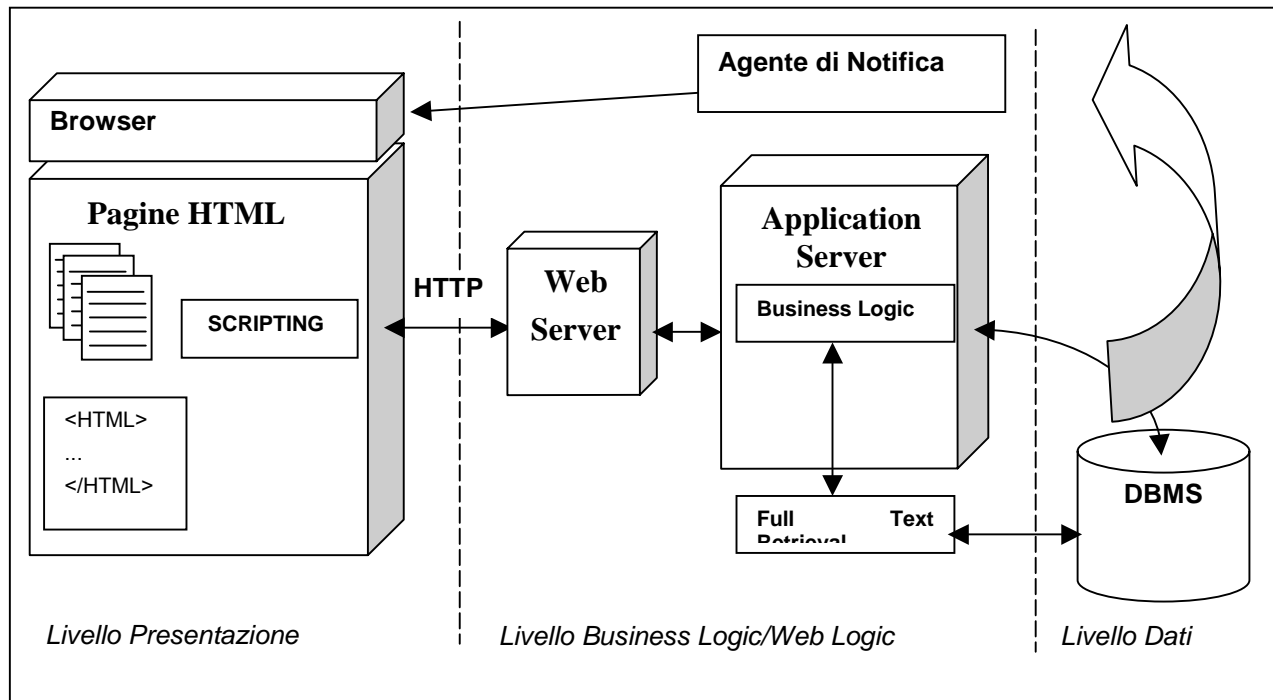
Disporre di informazioni in modo organizzato e tempestivo permette di acquisire i seguenti vantaggi:

- *Maggiore rapidità nella redazione di documenti legati al business della propria Azienda.*
- *Uniformità di impostazione nella produzione di documenti da parte di diverse persone a fronte di tematiche simili.*
- *Riuso di documenti precedenti.*
- *Addestramento più rapido dei neofiti.*
- *Capacità di individuare in modo certo le competenze presenti in Azienda su una determinata area operativa.*
- *Possibilità di accumulare e quindi diffondere la conoscenza specialistica creando delle "aree" di interesse in cui è possibile trovare esempi, FAQ, documentazione.*
- *Essere tempestivamente informati sulla natura delle informazioni prodotte.*

Il sistema ArchDoc offre attualmente una soluzione integrata a tali esigenze grazie ad una soluzione architetturale interamente basata sull'utilizzo di strumenti legati alle tecnologie Internet quali:

- *Sistemi di gestione documenti.*
- *Sistemi di ricerca su testi.*
- *Posta elettronica.*
- *Newsgroup.*
- *Messaging.*

- *Sistemi di gestione di basi dati.*



Il prodotto ArchDoc utilizza un'architettura logica a tre livelli:

- *Livello Presentazione*
- *Livello "Business Logic/Web Logic"*
- *Livello Dati*

Nel Livello di Presentazione ArchDoc fa uso, oltre all'HTML, di Java Script come linguaggio di scripting *client-side*.

Il livello Business Logic/Web Server viene implementato tramite *un qualsiasi Application Server che supporti la tecnologia dei Servlet Java*. La logica applicativa viene implementata all'interno di moduli denominati *Business Component* gestiti dall'Application Server. Attualmente ArchDoc utilizza PL/SQL come linguaggio per le *Business Component* e *Oracle Application Server 4.0.8.2* su piattaforma Windows NT per quanto riguarda il livello di *Business Logic/Web Logic*. Oltre ai *servlet* suddetti, Archdoc fa uso della tecnologia di *Full Text Retrieval* di Oracle (*InterMedia*) per implementare le funzionalità di ricerca testuale. All'interno dell'Architettura sono presenti degli agenti per la notifica automatica di eventi (es. avvertire gli utenti tramite *e-mail* della presenza, in base al proprio profilo, della presenza del sistema di un nuovo documento o per supportare le attività di classificazione dei documenti).

Il livello dati viene implementato tramite l'*RDBMS Oracle8i*.

Il sistema è dotato di meccanismi di sicurezza in base ai quali è possibile definire diversi livelli di policy per proteggere le diverse tipologie di documenti.

Gli utenti sono dotati di un profilo di abilitazione in base al quale è possibile:

- *Abilitare le aree tematiche/tipologie di documenti a cui si può accedere e definire quali funzioni (inserimento/modifica/cancellazione/ricerca/download) siano disponibili.*



- *Iscriversi al servizio di "Notifica di Eventi" scegliendo gli argomenti tematici per i quali si sottoscrive il servizio.*
- *Alimentare le varie aree tematiche mediante l'inserimento di domande, osservazioni, etc.*

Le informazioni sono organizzate all'interno del sistema per **tipo documento**. Gli utenti possono dinamicamente definire le proprie tipologie di documenti intendendo con ciò la possibilità di definire l'insieme degli attributi che ne costituiscono la struttura dati.

La struttura della Banca Dati prevede una gestione *multi-file* per ogni documento. Pertanto un documento è concepito come un'entità che può essere costituita da più file di formato e tipologia differente. Tutti i file sono fisicamente memorizzati all'interno dell'*RDBMS* Oracle in modo da assicurare meccanismi di *backup/recovery* controllato. Tutti i file indicizzabili sono automaticamente gestiti tramite l'opzione *InterMedia* di Oracle.

Per ogni tipologia di documento, il sistema genera in automatico le funzionalità per gestire le operazioni di:

- *Inserimento.*
- *Modifica.*
- *Cancellazione.*
- *Ricerca.*
- *Upload/Download File.*

Tutte le maschere sono automaticamente personalizzate in funzione della struttura dati definita per lo specifico documento.

L'insieme delle funzioni definite all'interno del sistema sono raggruppate per **Aree**, inoltre esiste il concetto di **Ruolo** inteso come l'insieme delle funzioni che possono essere assegnate in blocco ad un utente. Pertanto il meccanismo di sicurezza implementato nel sistema si basa sul concetto di "associazione utente/ruolo".

Altra funzionalità peculiare del prodotto è la possibilità di mantenere traccia di tutti gli utenti che consultano determinati documenti. Grazie a questa caratteristica è possibile conoscere chi ha consultato determinati documenti e quindi contribuire ad aumentare la rete di conoscenza interaziendale. Inoltre è possibile associare ad ogni documento e/o tipologia di documenti, l'elenco degli utenti e/o i ruoli aziendali che sono dei destinatari "istituzionali" del documento che si sta emettendo. In questo modo il sistema può, eventualmente, notificare agli utenti l'avvenuta pubblicazione del documento; inoltre l'Unità Responsabile della emissione dello stesso può periodicamente verificare l'elenco degli utenti che hanno effettivamente consultato il documento in modo da verificare che l'informazione sia stata effettivamente recepita dai destinatari.

Infine, il sistema di classificazione gestito dal prodotto può essere completamente personalizzabile dagli utenti in funzione delle specifiche esigenze delle diverse Unità Operative Aziendali.

## 4.2 Obiettivi di Engiweb

Il sistema Archdoc, così come è stato descritto nel paragrafo precedente, rimane al momento un prodotto di gestione documentale. Tuttavia, le funzionalità, già presenti, che consentono la definizione dinamica della struttura dei documenti e la generazione automatica delle relative primitive di gestione (inserimento, modifica/cancellazione, ricerca – anche *full-text*), può essere facilmente estesa al più generico concetto di informazione, al quale d'ora innanzi si farà riferimento.

Le numerose richieste di personalizzazione ed adattamento di Archdoc, provenienti dai vari clienti ai quali il sistema è stato presentato, consentono di delineare delle linee di sviluppo ulteriore, che in parte erano già allo studio.

- L'esigenza prioritaria appare quella di implementare dei **meccanismi di validazione** (la cui complessità potenziale va studiata con attenzione) dei contenuti informativi per il **web-**



**publishing.** Ciò consentirebbe l'utilizzo del sistema non solo come piattaforma di gestione documentale ma come infrastruttura di **Content Management** a supporto di siti web.

- Un altro obiettivo è costituito dalla definizione di un modello più sofisticato di accesso alle informazioni. L'attuale soluzione, che prevede l'utilizzo di utenti, ruoli e metodi relativi alle tipologie di informazione, consente, ad esempio, di verificare la possibilità che un utente del sistema possa effettuare un'operazione su tutti i documenti di una particolare tipologia. Da più parti si richiede la possibilità di scendere ad un livello di granularità più basso, per verificare la possibilità che *un singolo utente possa effettuare una determinata operazione su una singola istanza di informazione* (Es. diritti di modifica su un singolo documento, piuttosto che su tutti i documenti di una determinata tipologia, o rispetto a categorie semantiche di documenti).
- Ancora, importante appare la possibilità di definire assieme alla struttura dell'informazione da pubblicare, anche le *"regole di presentazione"* della stessa. Ciò vuol dire la possibilità di definire preventivamente quale *"elemento grafico"* debba essere utilizzato, piuttosto che il messaggio di help da associare ad un'istanza di informazione, etc.
- Un altro obiettivo è costituito dalla possibilità di definire all'interno del sistema un *workflow* delle informazioni, per guidare l'iter della stessa secondo le regole definite dall'organizzazione.
- Infine, una volta che vi sia la disponibilità per la produzione, la validazione, l'accesso controllato e la condivisione di informazioni di qualsiasi tipo, una utilizzazione naturale di questo genere di servizi potrebbe trovare riscontro, ad esempio, in un'applicazione di supporto alla gestione di progetti software, non solo in termini di documentazione, ma anche di codice sorgente e componenti software eseguibili. Engiweb sta già operando in questo senso per adattare l'attuale versione di Archdoc alla gestione di progetti *Open Source* interni.

Gli obiettivi sopra citati costituiscono un elenco non esaustivo delle possibilità di ulteriori sviluppi del sistema Archdoc. E' probabile che nuovi spunti vengano suggeriti da richieste di personalizzazione ed adattamento del sistema provenienti dal mercato. A questi vanno ad aggiungersi le attività di potenziamento e miglioramento delle funzionalità già disponibili, come ad esempio il supporto alla classificazione automatica delle informazioni.

Questi gli obiettivi dal punto di vista dei requisiti applicativi. Da quello della piattaforma tecnologica, invece, prioritario appare il *porting* del prodotto in *JSP*. Ciò consentirebbe di svincolare l'utilizzo del sistema dalla presenza dell'*Application Server* di Oracle, rendendolo disponibile su una piattaforma che rappresenta uno standard *de facto* e garantendo, con uno sforzo implementativo ragionevole, l'adattabilità del prodotto a qualsiasi sistema di *Server Pages*.

### 4.3 Obiettivi del Laboratorio di Ricerca e Sviluppo di Engineering

Il Laboratorio di Ricerca e Sviluppo di Engineering, da sempre avanguardia tecnologica della capogruppo, propone costantemente progetti di ricerca su scala europea volti all'investigazione e all'utilizzazione di nuove tecnologie.

Attualmente il laboratorio ha dato un nuovo impulso alla ricerca individuando tre aree di interesse per l'azienda. Accanto all'area del *Software Engineering*, nella quale il laboratorio ha lavorato fin dalla sua costituzione, sono state costituite quella di *Knowledge formalization and Management*, strettamente legata al business aziendale, e quella di *Human-Computer Interaction*, che raccoglie le sfide per la definizione di nuove soluzioni per il mercato del futuro.

#### 4.3.1 I progetti in corso

Nel quadro di queste attività, due dei progetti di ricerca co-finanziati dalla Commissione Europea<sup>9</sup> hanno evidenziato dei significativi punti di convergenza con alcune delle tematiche sin qui descritte. ECOLNET ed EUSLand vedono il laboratorio impegnato, rispettivamente, nello sviluppo

<sup>9</sup> <http://www.cordis.lu/IST>

di un modulo per la classificazione automatica ed uno per lo sviluppo di un motore di reasoning basato su mappe concettuali.

In particolare:

- \* **ECOLNET**<sup>10</sup>: è un sistema di *Knowledge Management collaborativo* il cui obiettivo è dare la possibilità alle piccole e medie imprese (PMI) di condividere informazioni e conoscenze relative al proprio mercato, per proporre nuove soluzioni di business nell'ambito europeo. L'interesse principale della C.E. che co-finanzia al 50% lo sforzo nella realizzazione del sistema è dato dalla possibilità per le PMI di poter affrontare i colossi del mercato europeo, senza necessariamente fondersi in grandi consorzi, senza perdere la propria identità e particolarità, semplicemente alleandosi con altre società in una ottica di business. L'architettura del sistema è altamente distribuita e basata su Internet. Ogni nodo (l'impresa) definisce in maniera completamente autonoma le informazioni e i dati da condividere (pubblicare) con il resto del network. Nell'ambito di questo progetto, Engineering, che ha la responsabilità della conduzione tecnica, sta sviluppando, tra l'altro due moduli relativi alla classificazione di testi e all'integrazione di sorgenti dati eterogenee e distribuite. Il progetto rappresenta anche la prima occasione di collaborazione tra Engineering e le alleate Ibermatica (Spagna) e Unilog (Franco-tedesca), membri fondatori di ESCAN<sup>11</sup>.
- \* **EUSLAND**<sup>12</sup>: è anch'esso un sistema di *Knowledge Management* nell'ambito di reti tematiche per pubbliche amministrazioni (PA) europee. Engineering, unico sviluppatore in senso stretto, ha il compito di fornire un framework per lo sviluppo di reti tematiche multilingua per P.A.. Questi moduli permettono la classificazione manuale di documenti in base ad una mappa concettuale (al momento si sta valutando la possibilità di usare Eurovoc<sup>13</sup> come base per i contenuti), e la ricerca delle informazioni guidata dalla semantica. Inoltre, l'architettura del sistema, consente la fruizione di questi servizi da applicazioni legacy (API CORBA) o attraverso il Web (API Servlet). Le iniziative di ricerca

E' interessante notare come le varie iniziative del laboratorio ricalchino sempre più gli interessi attuali delle controllate e partecipate del gruppo Engineering. In particolare, le proposte di ricerca afferenti all'area di *Knowledge Management and Formalization*, colgono appieno le necessità di futuri business della neo nata Engiweb.com.

Sul tema della classificazione il laboratorio ha in cantiere due lavori significativi: il primo relativo alla classificazione di documenti in lingua inglese, legato molto al progetto ECOLNET, mentre il secondo in maniera indipendente dalla lingua. La necessità di sperimentazione e valutazione di diversi approcci ed algoritmi per la classificazione di testi ha determinato la scelta di portare avanti lo sviluppo di moduli distinti che possono essere assemblati, secondo le esigenze del cliente al fine di integrare un classificatore per documenti completamente basato su tecnologia proprietaria.

L'interesse per la classificazione non si limita ai documenti di carattere testuale ma si estende al trattamento di altre forme di dati, come le immagini. Grazie alle competenze acquisite, presso la filiale di Palermo del Laboratorio, si esplorerà la classificazione di immagini, mentre è in fase di valutazione una proposta di progetto di Ricerca che intende proseguire gli studi sulla classificazione di componenti software<sup>14</sup>.

Sempre nella stessa area si hanno ulteriori spunti di ricerca, relativi alla reasoning su mappe concettuali (un lavoro che prenderà spunto da quanto realizzato in Eusland) e sull'integrazione

<sup>10</sup> European Collaboration Network (ref. <http://ecolnet.eng.it>)

<sup>11</sup> ESCAN è una società fondata a livello europeo dai presidenti delle tre società citate che ha l'obiettivo di estendere il mercato di Engineering all'ambito europeo in forma di alleanza commerciale con due importanti partners di ITC nell'area Iberica e dell'europa continentale. (ref. <http://www.escan-net.com>)

<sup>12</sup> EUropean System for Local Authorities'Networking Domains. (ref. <http://>)

<sup>13</sup> Eurovoc è un tesoro multilingua sviluppato per conto della C.E. che fornisce una categorizzazione dei termini usati nella documentazione della C.E. (ref. <http://europa.eu.int/celex/eurovoc/>)

<sup>14</sup> Attualmente il progetto Clarifi prevede lo sviluppo di una infrastruttura per lo scambio e la fruizione di componenti software. L'estensione proposta, Clarifi II, prevede il supporto alla classificazione di questi componenti.

delle informazioni del Web in una unica vista (una idea sulla quale si basa una nuova proposta di progetto di ricerca e che estendere gli obiettivi raggiunti con ECOLNET).

## 4.4 Sinergie

La missione del laboratorio, come spinta innovativa delle proposte al cliente, si concretizza nella ricerca di sinergie con le distinte anime del gruppo Engineering.

In particolare con Engiweb.com è stata identificata una *"intersezione comune"*, allo scopo di razionalizzare gli sforzi e consentire da un lato al Laboratorio di Ricerca e Sviluppo di poter utilizzare, già nei progetti in corso, una infrastruttura tecnologica e applicativa innovativa, dall'altro ad Engiweb.com di potenziare ed estendere le funzionalità applicative di WEB@WORK.

La compresenza negli stessi spazi fisici e le radici comuni hanno permesso di rimodellare le varie proposte di architettura dei sistemi di *Knowledge Management*, trovandone una modulare che permettesse di condividere gli obiettivi di sviluppo e soddisfare entrambe le esigenze.

### 4.4.1 I punti di contatto

Così come WEB@WORK, i progetti di ricerca citati nel paragrafo precedente, si basano sul web, hanno la necessità di definire dinamicamente le tipologie dell'informazione trattata, hanno necessità di formalizzare il patrimonio di "conoscenza" di un sistema introducendo dei meccanismi che estendano il concetto di criterio di classificazione di una determinata informazione e forniscano un supporto alla classificazione delle stesse.

In questo scenario la collaborazione tra Laboratorio e Engiweb.com si concretizza nello sforzo di entrambi alla convergenza dell'architettura su una base tecnologica comune e alla modularizzazione delle componenti sviluppate nei vari gruppi di lavoro. Il lavoro sinergico ha portato alla definizione e alla condivisione delle interfacce dei vari moduli mantenendo la struttura incrementale tipica di WEB@WORK.

### 4.4.2 I contributi del laboratorio

Alcuni dei componenti sviluppati dal Laboratorio saranno a disposizione di Engiweb.com per l'integrazione nelle verticalizzazioni di WEB@WORK proposte al cliente.

#### 4.4.2.1 Supporto alla classificazione

Il problema della classificazione di testi in linguaggio naturale è uno dei temi più trattati nell'ambiente della ricerca legato alla comprensione del linguaggio naturale.

L'esigenza di classificare un documento è comunemente risolta attraverso l'indicizzazione delle parole che fanno parte del testo.

Eventuali parole troppo comuni (coniunzioni, articoli) o non usati nella successiva fase di ricerca (avverbi) vengo eliminate attraverso le stop-list, elenchi di parole da non indicizzare.

La potenzialità di questo approccio (l'efficienza) è dovuta proprio alla mancanza di analisi e ragionamento sul contenuto del documento. Un documento è visto esclusivamente come una sequenza di caratteri, e una query è un pattern da identificare all'interno del testo stesso.

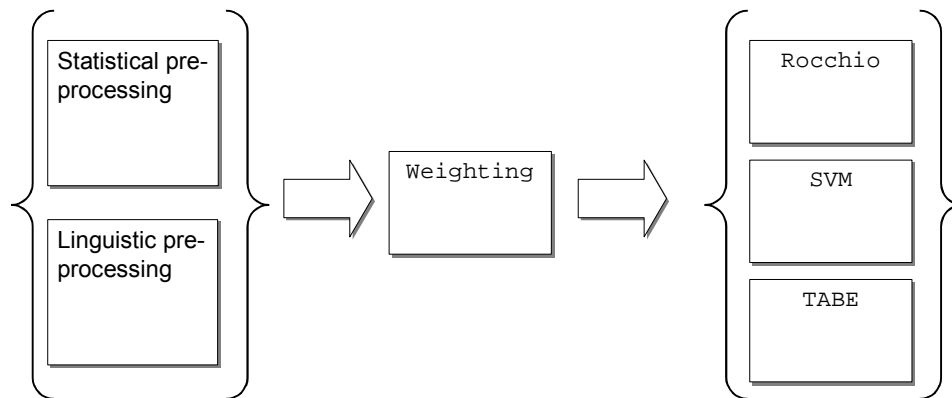
L'analisi lessicale (riduzione di più parole alla comune radice lessicale, l'analisi sintattica (riconoscimento di soggetti, verbi e complementi) e l'analisi terminologica (raggruppamento di parole contigue a formare un termine dall'unico significato), il raggruppamento di termini per stesso significato, sono alcune delle elaborazioni che si possono fare per migliorare la classificazione e, conseguentemente migliorare la precisione della ricerca.

Il laboratorio sta sviluppando (al momento si stanno effettuando i primi test funzionali) diversi tipi di classificatori: uno basato su un preprocessing linguistico (come descritto precedentemente) ed uno con pre-processing statistico.

In questo secondo approccio si vuole studiare la classificazione in maniera indipendente dalla lingua. Infatti l'analisi lessicale, sintattica e terminologica dipendono fortemente dalla lingua che si sta analizzando. Documenti in diverse lingue devono essere analizzati da differenti classificatori ognuno dipendente da una di esse.

L'approccio statistico permette, invece, di analizzare (e quindi ridurre il numero) i termini da indicizzare, basandosi solo su distribuzioni di probabilità che le stringhe assumono nel testo.

Entrambi i metodi prevedono una fase di *learning* che permette di calibrare i vari parametri presenti in base al dominio applicativo.



**Figura 1 Le possibili composizioni di moduli per il classificatore**

I sottomoduli che sono stati sviluppati sono: preprocessing linguistico per l'inglese, preprocessing statistico, modulo di pesatura linguistico, modulo di pesatura statistico e i moduli di classificazione basati sugli algoritmi Rocchio, SVM (*Support Vector Machine*) e TAE (*Textual Analysis Engine*). Tutti i moduli (in base alla loro funzione) saranno interscambiabili per ottenere configurazioni più adatte alle esigenze del progetto e del cliente finale.

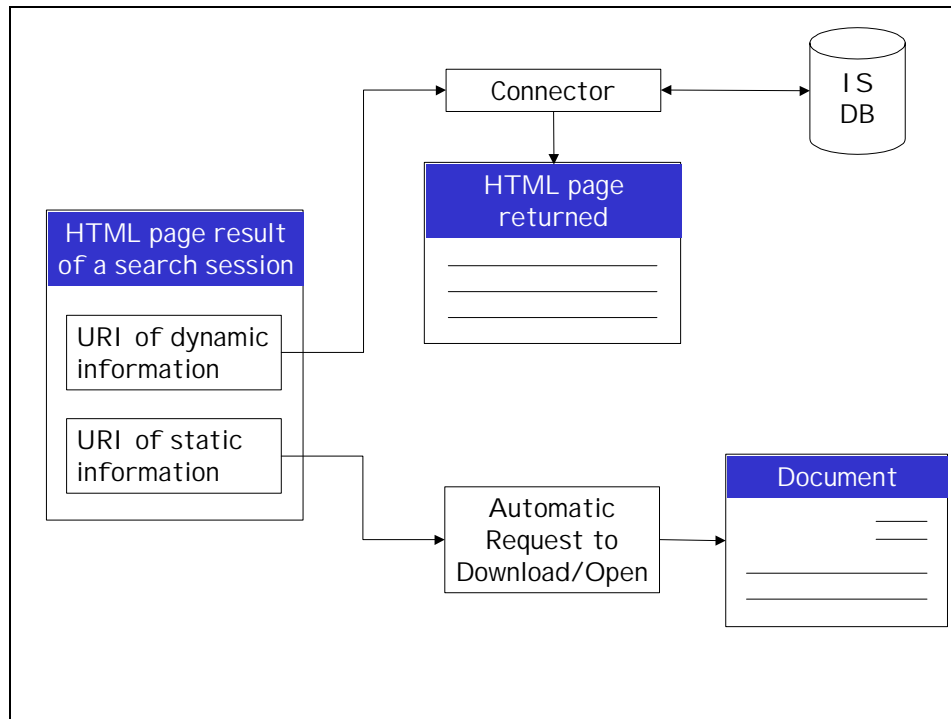
#### **4.4.2.2 Infrastruttura per lo sviluppo dei Connettori**

Molti dei prodotti per la gestione documentale non prevedono la possibilità di integrare nello stesso meccanismo di classificazione e gestione anche i dati presenti nei database.

Le informazioni, la conoscenza presente nei database, spesso viene persa, a meno di estrarre delle viste opportune di quei dati e generarne dei report. Questi report possono, poi, essere trattati come normali documenti. Tuttavia una grossa limitazione è data dalla necessità di aggiornare questi report ad un ritmo spesso troppo elevato.

La soluzione proposta nell'ambito del progetto ECOLNET prevede la definizione di una infrastruttura per la gestione di componenti software (connettori) in grado di accedere ai sistemi legacy dell'utente ed estrarre delle viste dei dati ivi memorizzati.

L'architettura è stata sviluppata completamente in Java e ogni modulo comunica con gli altri per mezzo di interfacce CORBA. In questo modo l'infrastruttura è indipendente dalla piattaforma e, potenzialmente, distribuita (per poter bilanciare i carichi di lavoro su differenti servers)



**Figura 2 Uniformità di gestione dei connettori e dei documenti in ECOLNET**

L'uso dei connettori consente di gestire nello stesso modo sia un documento che un insieme di dati su DB, attraverso una URI. Se ad un connettore si associa una descrizione testuale del risultato che dinamicamente produce, allora anche un connettore può essere classificato come un qualsiasi documento.

#### **4.4.2.3 Ricerche basate sulla semantica**

L'obiettivo del miglioramento della classificazione di informazioni è quello di ottenere dei risultati delle ricerche più conformi alle aspettative dell'utente.

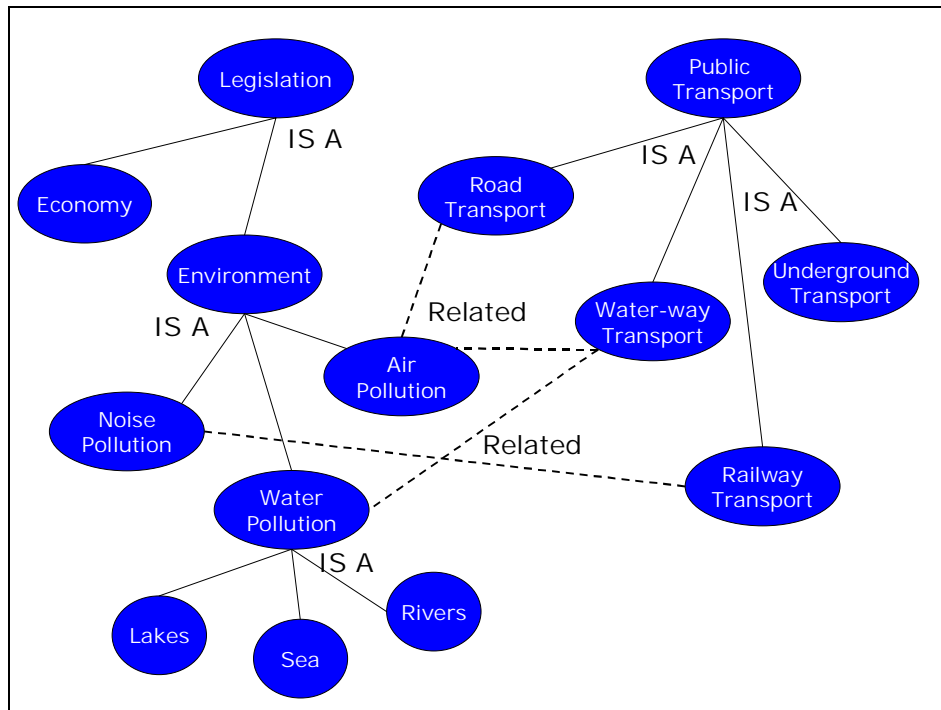
Molti degli attuali motori di ricerca sul web si basano su una indicizzazione delle stringhe contenute nelle pagine web. La richiesta di un utente viene tradotta nella ricerca della stringa di query all'interno delle pagine. Questo approccio, molto limitato produce spesso dei risultati non necessariamente congrui con l'ipotesi iniziale.

Si parla spesso di aggiungere semantica alle ricerche (nel caso precedente al Web).

Varie tecniche sono state proposte, ma quella che abbiamo considerato più promettente prevede l'uso di una mappa concettuale.

Mappa concettuale è un termine utilizzato nel campo della psicologia dell'apprendimento per descrivere i concetti di un ragionamento e i legami, le relazioni tra di essi.

Il laboratorio ha sviluppato un framework per la creazione di mappe concettuali, che viene utilizzato sia nella fase di classificazione (classificare divene allora associare ad ogni documento uno o più concetti) che nella fase di ricerca.



**Figura 3 Esempio di Mappa Concettuale utilizzata in Eusland**

Pertanto, nella ricerca si possono sfruttare anche le relazioni tra concetti, per restituire, non solo i documenti legati al concetto oggetto della query, ma anche quelli che, secondo certe relazioni, sono ad esso legati.

Al momento si stanno studiando diverse forme di ragionamento basandosi sulle tipologie principali di relazioni tra concetti.

Una interessante prospettiva dell'attuale uso delle mappe concettuali è quello della possibilità di effettuare interrogazioni in linguaggio naturale.

Analogamente a quanto accade con i documenti, una richiesta in linguaggio naturale dell'utente può essere "classificata" in base alla mappa concettuale ottenendo i documenti relativi alla classificazione. Questa soluzione che sembra promettente è, al momento, in fase di valutazione della precisione e delle performance.

#### **4.5 Evoluzione della specie: da Archdoc a WEB@WORK**

Nello sviluppo di applicativi web-based per una generica organizzazione, le richieste provenienti dal committente riguardano una serie di servizi applicativi dei quali si riporta di seguito a titolo esemplificativo, e senza alcuna pretesa di essere esaustivi, un elenco:

- "Chi è chi dell'organizzazione" (database delle anagrafiche e degli incarichi nell'organizzazione)
- Organigramma delle varie sedi dell'organizzazione (aggiornato dinamicamente)
- Calendario delle iniziative e degli eventi relativi all'organizzazione
- Gestione sondaggi interni ed esterni per acquisizione pareri
- Eventuale calendario concorsi per il personale interno (Intranet)
- Date di avvenimenti ufficiali
- Eventuali orari di apertura degli uffici dell'organizzazione
- Eventuale modulistica standard on line

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 23 di 80

- Eventuale fornitura di servizi riservati al personale dell'organizzazione o ai suoi clienti (prenotazioni, convenzioni, benefits ecc.)
- Visibilità delle eventuali procedure interne e Progetto Qualità dell'organizzazione
- Consultazione dati statistici circa la navigazione degli utenti
- bacheca News
- Forum di discussione
- Frequently Asked Questions
- ...

Tutti i punti dell'elenco precedente ineriscono il reperimento di dati, sia statici che dinamici, e la realizzazione di meccanismi di collaborazione. Anche se non esplicitati vanno poi considerati i servizi necessari alla produzione, validazione e pubblicazione dei dati di cui sopra e quelli che ne gestiscono il controllo di accesso.

E' quindi necessario astrarre, dalle specifiche richieste, gli insiemi di servizi generali che consentono l'implementazione di quelli poc'anzi elencati. Pertanto, più che il progetto di ogni singolo servizio ipotizzato dal committente deve essere delineata un'architettura generale per la realizzazione non solo dei punti di cui al precedente elenco, ma anche per la futura implementazione di nuovi servizi che dovessero rendersi necessari.

Si ritiene che l'obiettivo fondamentale nella realizzazione del complesso di servizi applicativi di una soluzione Intranet/Internet sia quello di costruire una infrastruttura **modulare e aperta**, volta a gestire al meglio il patrimonio informativo e ad implementare l'ambiente *collaboratory*, così come esso è stato definito in sede di descrizione del contesto tecnologico/organizzativo.

I requisiti della modularità e dell'apertura sono essenziali da un lato per una più razionale pianificazione della fase di progetto e realizzazione dei servizi applicativi richiesti, dall'altro per garantire una agevole aggiunta di altri servizi che in futuro si dovessero rendere indispensabili. E' evidente come questa possibilità dipenda non solo da una impostazione progettuale opportuna ma anche dall'adozione di una piattaforma tecnologica adeguata.

Se l'infrastruttura vagheggiata deve essere realizzata evolvendo in senso più generale Archdoc, appare evidente come la generalizzazione del contesto d'uso (non più soltanto gestione documentale, ma gestione di informazioni *tout court*, non più controlli di accesso alle risorse limitati ma estesi al massimo grado di flessibilità, non più implementazione di semplici regole di pubblicazione dell'informazione ma la possibilità di definire un vero e proprio *workflow*, etc.) imponga di modificare il nome da **ArchDoc** (**Arch**iviazione **doc**umentale) in qualcosa d'altro. Il nome proposto è **WEB@WORK** (l'etimo è evidente) .



## 5 L'architettura di WEB@WORK

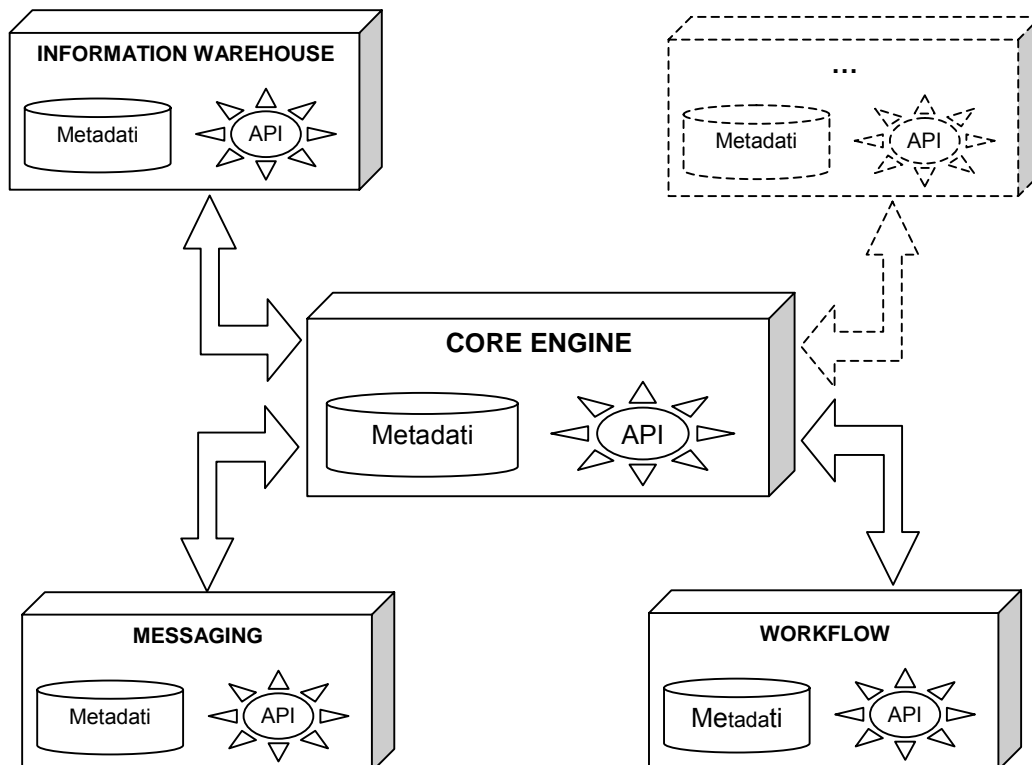
### 5.1 Panoramica

L'obiettivo di WEB@WORK è la creazione di una infrastruttura aperta e modulare per lo sviluppo di un ambiente di collaborazione web-based integrato e trasparente, dove il singolo strumento di lavoro perda la sua specificità per divenire parte di un ambiente virtuale di collaborazione.

WEB@WORK non è legato ad un uso specifico, ma al dominio di attività dei lavoratori e alle loro esigenze di comunicazione (sincrona, asincrona, video/grafica) e di informazione (gestione dei processi, documenti).

Attualmente il modello dei dati di Archdoc è in grado di supportare solo le funzionalità attualmente in essere. Qualsiasi nuovo requisito va soddisfatto estendendo il modello dei dati di un servizio applicativo esistente per supportare le nuove funzionalità. Se, ad esempio, si volessero introdurre dei meccanismi di controllo preliminare alla pubblicazione di un documento, si dovrebbero modificare le strutture dei dati che forniscono supporto alla gestione dei documenti (per gestirne, ad esempio, lo *stato*) e la parte applicativa per adattarle al nuovo requisito. Se il medesimo servizio dovesse essere reso disponibile per i messaggi di un forum di discussione (Es. validazione dei messaggi da parte di un moderatore) si dovrebbero modificare le regole di produzione e pubblicazione dei messaggi, anche qui intervenendo sulle strutture dei dati e sulla parte applicativa a supporto dei forum di discussione.

### 5.2 Architettura logica





L'architettura logica di *WEB@WORK* è caratterizzata, da un punto di vista topologico, da un "nocciolo" centrale di servizi (*Core Engine*) con il quale si integrano altri moduli che espongono ulteriori servizi applicativi più specializzati. Di ciascuno dei moduli individuati in prima istanza verrà data in seguito la descrizione dettagliata, mentre nei relativi documenti di analisi funzionale verranno descritti nel dettaglio i meta-modelli e le interfacce. La natura aperta e modulare del sistema determina la possibilità di aggiungere in futuro altri moduli per mettere a disposizione nuove classi di servizi applicativi specializzati.

### 5.3 Architettura applicativa e architettura operativa

L'architettura applicativa di *WEB@WORK* rappresenta un modello di riferimento che ha l'intento di evidenziare tutte le componenti che vanno realizzate e la separazione dei compiti tra esse.

In sintesi, i ruoli delle singole componenti sono:

#### **Canali di erogazione**

Supportano il colloquio con gli utenti finali attraverso diverse modalità di presentazione ed interazione. I canali di erogazione si differenziano in base al dispositivo di fruizione utilizzato e al protocollo di colloquio supportato.

#### **Gestione contenuti**

Supporta la redazione e la pubblicazione dei contenuti informativi.



#### **Sicurezza**

Supporta la protezione dei sistemi di erogazione.

La protezione si articola su più livelli:

- Protezione infrastrutturale: il controllo degli accessi da rete pubblica al sistema di erogazione (es. firewall, proxy ecc.)
- Protezione degli accessi degli utenti: il riconoscimento dello specifico utente in base alle credenziali presentate (es. password, smartcard, pin ecc.). I diversi sistemi consentono livelli di protezione distinti: la smartcard a differenza dell'uso di pin o password permette l'attuazione di meccanismi di autenticazione forte

- Utilizzo controllato dei servizi: consiste nella definizione di un profilo di abilitazione per ciascun utente dei servizi; tale profilo indica quali sono i servizi a cui l'utente può accedere dopo essere stato riconosciuto.

### **Servizi applicativi**

Supportano l'accesso a tutte le funzionalità del sistema informativo di *back-end* che si intende mettere a disposizione degli utenti.

### **Integrazione**

Supporta il dialogo tra la piattaforma di erogazione dei servizi ed altre eventuali sorgenti informative.

### **Applicazioni Legacy e Sorgenti Informative Esterne**

Possono essere presenti, nel contesto di una soluzione sviluppata con WEB@WORK, delle applicazioni *legacy*, che mettono a disposizione funzioni già esistenti in un dato sistema informativo.

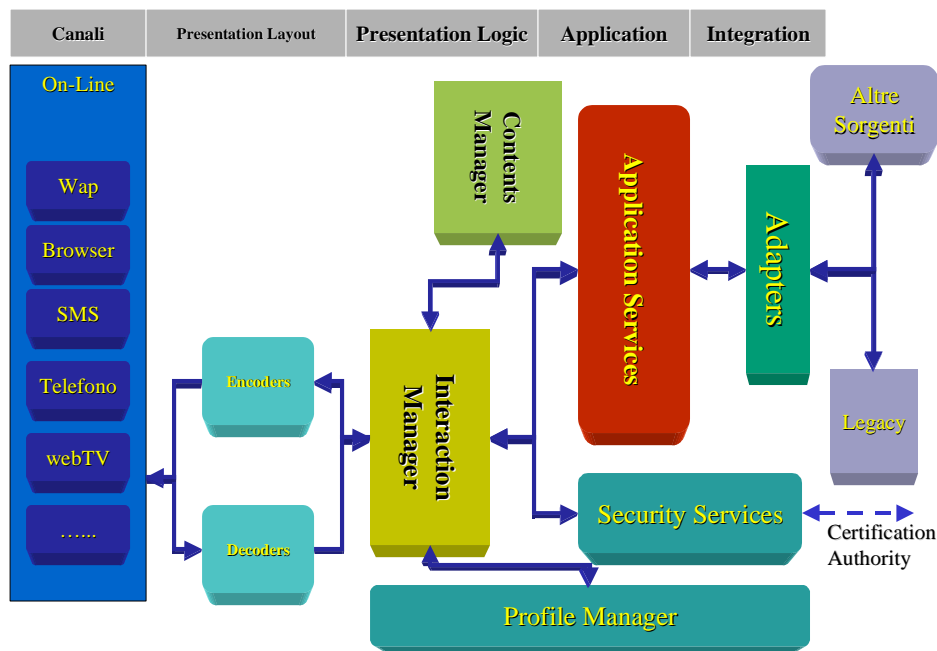
Alcune informazioni da erogare possono poi essere acquisite da altre sorgenti esterne e, mediante il meccanismo dei connettori, essere rese disponibili al pari di quelle gestite direttamente attraverso i servizi di di WEB@WORK.

### **Gestione Relazione Utente**

Supporta il trattamento dei dati rilevati nei rapporti intercorsi con gli utenti di un sistema.

Tale supporto consiste nel raccogliere i dati relativi all'utilizzo dei servizi e al mantenimento di un database dei contatti avuti per poter personalizzare il rapporto con un dato utente.

L'architettura applicativa data nella figura precedente si traduce nei seguenti moduli rappresentati nella figura che segue.

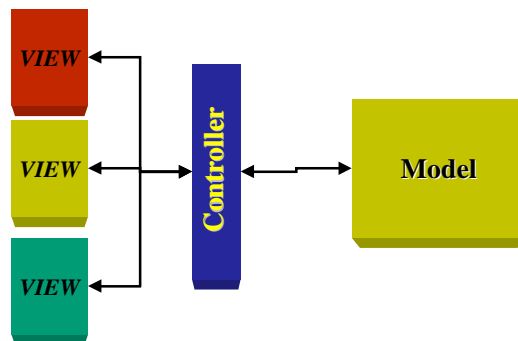


Essa rappresenta l'architettura operativa di WEB@WORK in termini di moduli e loro interrelazioni e si rifà al paradigma noto in letteratura col nome *Model-View-Controller (MVC)*. Tale paradigma impone che l'architettura sia suddivisa in tre layers:

- **View** è il *layer* che gestisce la presentazione delle informazioni all'utilizzatore finale di un servizio erogato attraverso un qualsiasi canale tecnologico.
- **Controller** è il *layer* che governa la logica dell'interazione fra utente e servizio erogato:
- **Model** è il *layer* che implementa la logica applicativa (le funzioni). Tale logica deve essere assolutamente sganciata dai layers sopra descritti per massimizzarne la condivisione tra i diversi canali e per rendere i singoli servizi applicativi indipendenti (in termini funzionali e di dati trattati) dallo specifico canale di erogazione.

Come si vede dalla figura seguente, ad un *Model* (funzione) possono corrispondere più *View* (presentazioni). Tale paradigma permette di mantenere separata la logica applicativa di una data funzione dalle modalità con cui uno specifico utente interagisce con essa.

Tale separazione permette di modificare le funzioni senza dover modificare necessariamente la presentazione e di costruire una nuova presentazione (ad esempio per un dispositivo di fruizione con caratteristiche peculiari) per una funzione esistente senza dover modificare quest'ultima.



WEB@WORK applica il paradigma MVC separando le componenti architetturali nei seguenti layers:

- *Presentation Layout Layer* in cui è implementata la gestione della presentazione
- *Presentation Logic Layer* in cui è gestita la logica di interazione;
- *Application Layer* in cui è implementata la logica applicativa;
- *Integration Layer* in cui è implementata la logica di integrazione.

Lo schema di riferimento è analogo a quello di Archdoc, con opportune estensioni.

WEB@WORK è un'infrastruttura di servizi applicativi caratterizzata da un meta-modello di dati e da diverse API che espongono i servizi. Il livello di Presentazione di WEB@WORK è quindi costituito unicamente da un'interfaccia web di amministrazione del sistema. In casi particolari anche il *front-end* di una soluzione web basata su WEB@WORK può essere gestito con

un'interfaccia ricavata da quella di amministrazione. Più in generale, però, la singola applicazione web (ad esempio un sistema di gestione di documenti, un sistema di pubblicazione di notizie su un sito, etc.) costituirà una *verticalizzazione* opportuna di WEB@WORK, utilizzandone solo una parte dei servizi forniti.

Da ciò deriva che dal punto di vista più meramente tecnico-architettuale, le macrofunzionalità, elencate nel precedente paragrafo, si traducono in altrettanti sottosistemi che indicheremo con i nomi convenzionali di

- **Encoders/Decoders**
- **Interaction Manager**
- **Contents Manager**
- **Application Services**
- **Security Services**
- **Profile Manager**

I sottosistemi evidenziati nella figura vengono descritti nei paragrafi che seguono.

### 5.3.1 *Encoders/Decoders e Interaction Manager*

Interaction Manager ed Encoders/Decoders si occupano della presentazione dei contenuti e dell'interazione con l'utente finale/amministratore.

La presentazione e l'interazione sono condizionate da diversi fattori:

- Limitazioni intrinseche del canale e del dispositivo di fruizione utilizzato,
- Profilo dell'utente a cui le dette informazioni sono dirette;
- Regole che governano la personalizzazione del servizio continuamente aggiornate.

WEB@WORK tiene conto di questi condizionamenti secondo quanto di seguito descritto.

I contenuti presentati risultano essere l'aggregazione di diverse parti ciascuna delle quali può avere una propria struttura e può essere trattata separatamente.

Inoltre, come si è visto la presentazione dipende da diversi fattori che variano nel tempo.

Per questo motivo, WEB@WORK deve fornire dei servizi applicativi che, relativamente alle possibili verticalizzazioni, consentano la costruzione dinamica della presentazione delle informazioni in funzione della struttura dei contenuti da presentare, del profilo dell'utente, del dispositivo di fruizione che l'utente utilizza.

*Interaction Manager* governa l'assemblaggio dinamico delle componenti che costituiscono l'oggetto da visualizzare all'utente finale; tale assemblaggio è guidato dalle indicazioni provenienti dal profilo dell'utente.

I contenuti erogati attraverso i servizi di WEB@WORK vengono forniti con una rappresentazione XML (eXtensible Markup Language)<sup>15</sup>, che consente di isolare il contenuto dalle modalità di rappresentazione, e presentati sotto diverse forme in funzione del canale e del dispositivo d'accesso. *Encoders/Decoders* si occupano di gestire tale eterogeneità.

### 5.3.2 *Contents Manager*

**Contents Manager** è il sottosistema che gestisce la memorizzazione dei documenti in formato elettronico strutturato. I servizi applicativi saranno quelli del modulo logico "*Information Warehouse*" in precedenza introdotto. Il database sarà strutturato in modo tale da permettere la fruizione dei documenti in esso contenuti attraverso un'interfaccia web.

In particolare Contents Manager:

<sup>15</sup> XML, così come descritto nella W3C Recommendation del 10/02/1998, codifica un metalinguaggio che permette di definire linguaggi a marcatori per specifici domini applicativi.

Fornisce tutte le funzioni di supporto ai redattori dei contenuti. La redazione dei contenuti può essere fatta con diversi strumenti esterni all'ambiente. I contenuti redatti vengono memorizzati in un repository. Dopo la memorizzazione i contenuti attraversano un processo di approvazione supportato automaticamente e definibile da utenti con profilo opportuno. Al termine di tale processo i contenuti vengono portati nello stato di pubblicazione e sono richiamabili dai template che gestiscono la presentazione grafica eseguiti da *Interaction Controller*.

Si noti che la separazione fra presentazione e logica applicativa adottata da WEB@WORK e la costruzione dinamica della presentazione in funzione del profilo tecnografico dell'utente permettono al sistema di erogare gli stessi contenuti anche verso dispositivi diversi dal browser (cellulari WAP, PDA ecc.) modificando il solo *layer* di presentazione e riducendo al minimo gli impatti sulla logica applicativa.

Per quanto riguarda la gestione dei documenti è da considerare che *Contents Manager* si deve tenere conto che i documenti possono essere di formato eterogeneo, per cui il repository dovrà gestire tale eterogeneità.

A tutti i documenti memorizzati potrà essere associato un *tipo*, espresso da un insieme di attributi che possono essere definiti dinamicamente. Pertanto, l'insieme di attributi associati al documento è dinamico e variabile nel tempo. La funzione di indicizzazione opera sia sugli attributi del documento che sul contenuto in modo da permettere l'implementazione di ricerche sia per attributo che per testo.

Contents Manager metterà a disposizione due tipi di ricerca:

*Ricerca per attributo* che ricerca tutti i documenti che hanno valori degli attributi che soddisfano le condizioni poste; poiché gli attributi associati ai documenti sono definiti dinamicamente anche la funzione di ricerca supporta tale dinamicità;

*Ricerca testuale* che ricerca tutti i documenti che contengono un determinato testo.

### 5.3.3 Adapters

**Adapters** è la componente che gestisce il colloquio fra il sottosistema di erogazione sui canali, l'eventuale sistema legacy e le eventuali altre sorgenti informative.

L'integrazione delle applicazioni residenti sul sistema legacy è resa complessa dall'eterogeneità delle modalità di dialogo e della struttura dei dati scambiati.

Per ridurre tale complessità, il problema dell'integrazione è stato scomposto in tre layers:

- *Low layer* che implementa la gestione delle modalità di comunicazione, intendendo con ciò i protocolli di base che l'eventuale sistema legacy da integrare mette a disposizione; dato che le applicazioni legacy risiedono tutte su sistema mainframe, l'insieme di protocolli di base da interfacciare è costituito dalla suite SNA. Pertanto, l'architettura non implementa una soluzione propria, ma si appoggia su gateway software di mercato che supportano l'intera suite SNA.
- *Middle layer* che implementa la gestione del dialogo con le applicazioni, intendendo con ciò le regole che governano la struttura dei dati scambiati, l'invio delle richieste e la ricezione delle risposte;
- *High layer* che implementa la gestione della semantica della comunicazione che inerisce le modalità con cui si dialoga con le singole funzioni legacy. In esso si concentrano i meccanismi per attivare una funzione, passare ad essa i parametri di input, gestire il suo output ecc.

Gli Adapters implementano i servizi del *middle* e *high* layer, rendendo trasparente al sistema di erogazione dei servizi sui canali tecnologici le peculiarità tecniche dell'accesso alle applicazioni residenti eventualmente su host.

In particolare, gli Adapters disaccoppiano il dialogo fra l'infrastruttura di erogazione dei servizi e il sistema legacy. Le funzioni principali di un Adapter sono:

*Trasmissione messaggi eterogenei*

Consiste nel prendere in carico un file da una sorgente e portarlo fino alla destinazione stabilita. Il servizio di trasmissione si occupa non soltanto del trasporto delle informazioni ma anche della qualità del servizio : consegna garantita e fault-tolerance

*Trasformazione di formato*

è servizio di data mapping che in base a regole precodificate per ciascuna tipologia di messaggio (o dato elementare) trasforma l'informazione dal formato nativo in un formato di destinazione.

*Istradamento automatico*

È il servizio che si occupa governare l'istradamento di un messaggio verso una determinata destinazione in base a regole predefinite che ineriscono il contenuto del messaggio

*Controllo della trasmissione*

È il servizio che raccoglie tutti dati sulle attività svolte dagli altri servizi (trasmissione, trasformazione ed istradamento). Emette delle notifiche al verificarsi di eventi significativi.

### 5.3.4 Security Services e Profile Manager

**Security Services** è l'insieme di servizi che governa l'accesso al portale ed il monitoraggio delle attività.

La sicurezza complessiva di un sistema è un valore dipendente da due fattori: l'infrastruttura tecnologica utilizzata per implementare i diversi servizi di sicurezza e la policy di sicurezza applicata.

I due fattori sono interdipendenti, infatti, la loro sommatoria dà la misura del rischio intrinseco correlato erogati attraverso una rete ad accesso pubblico.

La loro interdipendenza si esplica nel fatto che le eventuali debolezze dell'infrastruttura tecnologica devono essere compensate da una policy di sicurezza che tuteli l'erogatore del servizio dagli effetti dell'uso fraudolento degli stessi.

Si noti che un'infrastruttura tecnologica di sicurezza ipoteticamente inattaccabile può comportare dei costi che, in molti casi, non hanno un ritorno tale da giustificarli, pertanto le eventuali debolezze dell'infrastruttura tecnologica possono essere il risultato ponderato di un'analisi costi-benefici.

In accordo a quanto stabilisce l'organismo internazionale di standardizzazione, International Standard Organization (ISO), vi sono cinque servizi di sicurezza che un'infrastruttura per sistemi basati su reti ad accesso pubblico deve offrire:

- *Identificazione e Autenticazione*, che deve determinare in modo inequivocabile l'agente che sta richiedendo l'accesso ad una risorsa e garantire al destinatario della richiesta l'identità del richiedente.
- *Controllo degli Accessi*, che deve definire a quali oggetti del sistema (dati o applicazioni) l'agente autenticato può accedere.
- *Riservatezza*, che deve far in modo che soltanto il destinatario di una data informazione possa leggerla.
- *Integrità*, che deve permettere all'agente destinatario di una richiesta di poter verificare con certezza che il contenuto della richiesta stessa non sia stato alterato;

- *Non Ripudio*, che deve far in modo che né il mittente né il destinatario di un messaggio possano negare di avere eseguito l'azione.

Per soddisfare i livelli suddetti *Security Services* deve mettere a disposizione le seguenti funzionalità.

#### *Autenticazione*

Servizio che autentica l'utente che si collega al sistema in base alle sue credenziali. Le credenziali possono essere di diverso tipo, nel contesto delineato dalla richiesta d'offerta è possibile differenziare il livello di credenziali richiesto a seconda dell'area del portale in cui l'utente intende entrare.

<i>Livello</i>	Identificazione	<i>Operatività</i>
1	Credenziali: user e password	Autenticazione debole, dovrebbe consentire l'accesso ad aree destinate a tutti gli utenti e/o ad aree che propongono contenuto con basso livello di riservatezza
2	Credenziali: certificato SSL	Autenticazione della postazione, dovrebbe consentire l'accesso ad aree protette e destinate a gruppi di utenti specifici che propongono contenuto con basso livello di riservatezza
3	Credenziali: certificato di firma	Autenticazione forte basata sul meccanismo del challenge-response che utilizza il certificato digitale personale di firma; l'utente può avere accesso alle aree riservate e con elevato contenuto di riservatezza

#### *Single sign-on*

Il single sign-on fa sì che un utente autenticato possa accedere a diverse applicazioni durante la stessa sessione senza dover immettere di nuovo le credenziali.

#### *Protezione dei dati scambiati*

La protezione dei dati scambiati è una funzionale ortogonale a tutti i processi.

In funzione del processo e del documento trasmesso si possono attuare livelli di riservatezza ed integrità diversi.

È possibile attuare diversi livelli di protezione dei dati trasmessi a cui corrispondono tecniche diverse.

Con la cifratura del canale (sia essa SSL che tramite tunneling) si ottempera al requisito di integrità.

Cifrando il documento da trasmettere si ottempera al requisito di riservatezza.

#### *Profile Manager*

Supporta la gestione da parte dell'amministratore dei profili abilitativi. I profili abilitativi descrivono che cosa un utente può fare e a quali risorse può accedere.



A ciascun utente deve essere associato un profilo abilitativo che definisce le attività che può svolgere e gli oggetti a cui può accedere. Al termine del processo di autenticazione l'utente è ed è quindi possibile individuare il suo profilo abilitativo che può variare anche in funzione delle credenziali che ha presentato per autenticarsi come illustrato nella tabella sopra data. Il profilo abilitativo deve essere costruito ed aggiornato attraverso funzionalità di amministrazione utilizzate dall'amministratore del sistema.

## 6 WEB@WORK: meta-modello e interfacce funzionali

Per ottenere una architettura modulare come quella descritta precedentemente è necessario *unificare e normalizzare i servizi di base all'interno di un modello dei dati e di una API comuni e coerenti*. In ArchDoc ogni servizio applicativo utilizza il proprio schema di *metadati* e *"mappa"* questi dati con oggetti dell'applicazione (messaggi di un forum piuttosto che documenti, etc.). Unificare e normalizzare tali interfacce consente di razionalizzare la logica delle applicazioni minimizzando le ripetizioni di codice.

### 6.1 Un modello ad oggetti

L'insieme di *metadati* generale, in grado di supportare qualsiasi entità all'interno del sistema, viene rappresentato da un *"modello ad oggetti"*. Con il termine *metadati* intendiamo qui l'insieme dei dati necessari all'abilitazione di certi servizi generici, al di fuori dei singoli modelli di dati dei moduli che forniscono gli altri servizi applicativi. Con il termine *"oggetto"* ci si riferisce qui a *qualsiasi entità rappresentata all'interno del sistema* (documenti, informazioni, utenti, gruppi, permessi, etc.).

Il modello ad oggetti che viene proposto è caratterizzato dai seguenti requisiti fondamentali:

- *Possibilità di identificare univocamente un oggetto.* Per mettere a disposizione dei servizi applicativi generali è necessario che tutti gli oggetti dell'applicazione possano essere identificati in maniera non ambigua. Ad esempio, attualmente, Archdoc utilizza sovente la tupla di valori (ID\_RUOLO, ID\_FUNZIONE, ID\_UTENTE) per determinare l'insieme di permessi che un utente ha all'interno del sistema. Chiavi *"composte"* come la precedente identificano implicitamente degli oggetti, definendo degli ID generali per composizione di diversi *"pezzi"* del modello di dati. Il problema è che la loro definizione ed utilizzo sono realizzati di volta in volta e che non c'è consistenza tra le diverse tuple. Ciò rende difficile la realizzazione di servizi generici indipendenti dall'applicazione. *WEB@WORK* attribuisce perciò un identificatore univoco in maniera centralizzata a qualsiasi oggetto all'interno del sistema.
- *Controllo unificato degli accessi.* Il controllo dei meccanismi di accesso agli oggetti dovrebbe essere il più possibile trasparente ai moduli applicativi. Sinora ciascun modulo applicativo doveva gestire il controllo di accesso ai suoi dati per proprio conto (Es. Nel modulo di gestione documentale il controllo di accesso alle tipologie di documento veniva gestito da strutture di dati che mettevano in associazione il tipo documento con l'utente o con il ruolo, nel modulo Forum analoghe strutture associative tra messaggi e ruoli particolari – come il moderatore del forum – venivano gestite tramite un'informazione circa lo *"stato"* del messaggio, etc.). *WEB@WORK* supporta un più generale sistema di controllo degli accessi che consente *"ambiti di appartenenza"* gerarchici. Dove per ambito di appartenenza va inteso il contesto di sicurezza al quale il generico oggetto appartiene (Es. Associazione di un'informazione ad un utente, ad un gruppo o a tutti).
- *Modello dei dati aperto e dinamicamente estendibile.* Il modello ad oggetti di *WEB@WORK* consente agli utenti di definire una gerarchia *classi di oggetti* con meccanismi di *derivazione di sottoclassi* ed *ereditarietà*. Gli sviluppatori potranno consentire agli utenti la definizione dinamica delle strutture delle informazioni trattate, tuttavia mentre ciò è attualmente realizzato senza modificare il modello dei dati sottostante, in *WEB@WORK* la definizione di nuove classi di oggetti comporterà l'aggiunta automatica di nuove strutture di dati per la memorizzazione delle loro istanze.



- *Relazioni tra oggetti.* **WEB@WORK** deve fornire supporto alla definizione di generiche relazioni tra oggetti. Ciò viene fatto consentendo agli sviluppatori di definire una speciale tipologia di oggetti, chiamata “*classe relazione*”. Questa è essa stessa una *classe di oggetti* che non fa nulla salvo che rappresentare il concetto di relazione.
- *Gestione automatica del livello fisico dei dati.* Le API del modello ad oggetti di **WEB@WORK** debbono consentire la generazione automatica di strutture di dati fisiche (tabell, indici, vincoli di integrità referenziale, etc.) all'interno del database a supporto delle classi di oggetti e delle classi di relazione definite all'interno del sistema, rendendo del tutto trasparenti i meccanismi di ereditarietà definiti tra classi di oggetti.

## 6.2 La piattaforma tecnologica e il modello dei dati

**WEB@WORK**, come il suo predecessore, utilizza come piattaforma tecnologica un Database Oracle 8i, utilizzando solo il livello “*relazionale*”, per ciò che riguarda dati e metadati, e il PL/SQL come linguaggio per realizzare le *API* sia del *core engine* che degli altri moduli applicativi.

Al di là delle giustificazioni, pure importanti, riguardanti la diffusione di queste tecnologie sul mercato e le competenze presenti su di esse all'interno dell'azienda, la presenza di una infrastruttura che consenta la derivazione per estensione di classi di oggetti e fornisca meccanismi di ereditarietà potrebbe sollevare la questione se non sia più giusto utilizzare un database ad oggetti.

Il motivo principale consiste nel fatto che la maggior parte dei database ad oggetti disponibili sul mercato sono legati ad un qualche linguaggio di programmazione orientato agli oggetti. L'idea è sempre quella di fornire una integrazione a livello del linguaggio di programmazione per minizzare la “*distanza*” tra il database e il linguaggio stesso. Quindi gli oggetti e le classi del database sono modellati, in genere, direttamente a livello di oggetti e classi del linguaggio. Ciò rende difficoltoso, e a volte impossibile, interagire con tali database attraverso un linguaggio che non abbia con essi un così alto grado di “*accoppiamento*”.

Vi sono database, e Oracle è tra questi, che mettono a disposizione un modello misto (*Object Relational*) fornendo così un'interessante alternativa. Qui una qualche nozione di astrazione di *classe* e di meccanismi di ereditarietà sono inclusi all'interno del motore SQL tramite opportune estensioni. Il problema consiste nel fatto che tali estensioni sono implementate in modo proprietario dai diversi DBMS. Inoltre la maggior parte di questi sistemi presentano dei vincoli sintattici e operativi che rendono l'utilizzo dei meccanismi di ereditarietà difficile in pratica.

In conclusione, la soluzione migliore appare quella di aggiungere quelle caratteristiche orientate agli oggetti di cui abbiamo bisogno tramite opportune strutture di metadati, continuando però a trarre profitto dalla generalità di un modello dati relazionale. Nel contesto di **WEB@WORK** ciò significa utilizzare un modello ad oggetti per rendere il modello dei dati più flessibile, così che nuovi moduli applicativi possano accedere a servizi generali. Tuttavia, tale modello ad oggetti, per come è stato disegnato, *non è pensato per la memorizzazione su larga scala dei dati delle varie applicazioni, ma solo per la rappresentazione e la memorizzazione dei metadati.*

## 6.3 La piattaforma tecnologica e l'ambiente di sviluppo delle API

L'utilizzo del linguaggio PL/SQL per la realizzazione delle API sia del *core engine* che dei moduli applicativi che con esso si integrano è dettata da alcune considerazioni fondamentali.

Il codice PL/SQL è memorizzato in forma compilata all'interno del database Oracle. Questo garantisce dei livelli di prestazione ottimali, soprattutto, come è evidente, per tutte le operazioni che accedono pesantemente ai dati. Le versioni 8.1.x di Oracle includono una *Java Virtual Machine* all'interno del database, che consente di realizzare applicazioni che accedano ai dati direttamente in Java. In ogni caso l'utilizzo di classi Java deve essere effettuato definendo delle procedure PL/SQL che fungano da *wrapper* del codice Java, quindi PL/SQL costituisce in ogni caso l'unico linguaggio di interfaccia per il codice memorizzato all'interno del database.

<u>Stato del Documento</u>	<u>Rif.</u>	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 34 di 80

Va considerato, inoltre, che l'invocazione di *stored procedures* PL/SQL è possibile pressoché da qualsiasi ambiente di programmazione esterno (ODBC, JDBC, Gateway Proprietari relativamente a prodotti specifici, etc.).

Il codice PL/SQL è per definizione *portabile* ovunque lo sia il DBMS. Il medesimo codice opera senza necessità di alcuna modifica su tutte le piattaforme su cui il DBMS Oracle sia disponibile. Esso, essendo memorizzato all'interno Database, è sottoposto alle medesime politiche di backup e recovery, con ovvi vantaggi dal punto di vista della consistenza e della robustezza delle applicazioni.

## 7 Il “Core Engine”

Come descritto in precedenza nel paragrafo relativo all'architettura del sistema, il componente principale di *WEB@WORK* è il *Core Engine*, il cui scopo è quello di fornire dei servizi applicativi generali, utilizzabili da tutti i moduli che si integreranno poi nel sistema. I due aspetti fondamentali che vengono implementati nel *core engine* sono il *modello ad oggetti* poc'anzi descritto e il sistema di sicurezza, per la gestione centralizzata del controllo di accesso alle risorse.

### 7.1 I Metadati di supporto al modello ad oggetti

#### 7.1.1 Analisi dei requisiti

Di seguito vengono elencati e descritti i requisiti del modello ad oggetti, sia in termini delle strutture di dati utilizzate, che in quelli di definizione di un'interfaccia di programmazione che esponga i relativi servizi applicativi.

##### 7.1.1.1 Requisiti del modello dei dati

Nel paragrafo relativo all'introduzione del modello ad oggetti del sistema, sono stati definiti alcuni requisiti fondamentali, che brevemente si richiamano:

- *Identificazione univoca degli oggetti all'interno del sistema*
- *Controllo unificato degli accessi*
- *Modello dei dati aperto e dinamicamente estendibile*
- *Supporto alla definizione di relazioni tra oggetti*

Il fatto che ogni oggetto abbia un identificatore unico all'interno di *WEB@WORK* significa che il *core engine* può prendere in carico qualsiasi oggetto all'interno del sistema in maniera generica. In tal modo l'unica azione che si richiede ad un modulo applicativo per ottenere dei servizi generali dal *core engine* è quello di *agganciarsi* al sistema ad oggetti. Questo espone gli oggetti con un ID univoco e con le informazioni su quali siano le strutture di dati che memorizzano i dati dell'applicazione associati agli oggetti.

L'identificatore di un oggetto può essere utilizzato per riferirsi a collezioni di dati applicativi eterogenei. Ancora più importante, gli ID degli oggetti consentono agli sviluppatori di sviluppare ed utilizzare rapidamente dei servizi applicativi generali.

Una *classe di oggetti*, così come introdotta nel §3.1, si riferisce alla specificazione di una o più proprietà per gestire i dati applicativi. Il modello ad oggetti di *WEB@WORK* fornisce delle strutture di dati per descrivere e rappresentare le classi. Qualcosa di analogo c'è in Archdoc, laddove è possibile definire delle tipologie di documento definendo gli attributi opzionali che un tipo di documento utilizza, oltre agli attributi generici, comuni a tutti i documenti, indipendentemente dalla tipologia. Il modello ad oggetti di *WEB@WORK* consente una generalizzazione di questo meccanismo. Il modello dei dati del *core engine* deve essere in grado di rappresentare classi di oggetti con queste caratteristiche:

- *Nome della classe* (con vincolo di unicità).
- *Proprietà della classe* (i cui valori sono condivisi da tutte le istanze della classe. Il nome della proprietà deve essere univoco nell'ambito di una determinata classe).
- *Proprietà dell'oggetto* (specifiche per il particolare oggetto della classe).

Il modello ad oggetti deve supportare la definizione di classi che siano sottoclassi (per estensione) di classi esistenti. Una sottoclasse eredita tutti le proprietà della classe da cui è derivata, oltre a definirne di peculiari. Un aspetto critico del modello ad oggetti è costituito dal fatto che le classi possono essere “modificate” e che tali modifiche debbano propagarsi a livello delle sottoclassi derivate. Relativamente alle sottoclassi, il modello ad oggetti determina dei vincoli analoghi a quelli descritti in precedenza (unicità del nome della sottoclasse e delle

proprietà) con in più quello che i *nomi delle proprietà estese in una sottoclasse non devono essere in conflitto con quelle di una superclasse che la preceda nella gerarchia*.

Il modello ad oggetti di *WEB@WORK* fornisce la possibilità di definire dei *metodi* relativamente alle classi di oggetti. Un metodo è del codice procedurale che può agire su tutte le istanze di una classe.

Oltre alle informazioni sulle classi, viene fornita una memorizzazione centralizzata dei valori delle proprietà degli oggetti. Le uniche informazioni necessarie per recuperare il valore di una proprietà di un oggetto devono essere l'identificativo univoco di quest'ultimo (ID) e il nome della proprietà.

Il sistema consente il recupero automatico dei valori delle proprietà ereditate da una superclasse per un oggetto appartenente ad una sottoclasse e permette allo sviluppatore di definire dei vincoli sui valori che una proprietà può assumere, allo scopo di mantenere specifiche regole di integrità sui dati dell'applicazione.

Relativamente ai servizi di controllo unificato degli accessi, *WEB@WORK* mette a disposizione una nozione generale di "*ambito*", che mette lo sviluppatore in grado di definire una gerarchia di "*contesti*" di validità degli oggetti. Questi *contesti* sono alla base del sistema di permessi di *WEB@WORK*. In generale, se ad un oggetto non sono associati permessi espliciti, esso eredita l'insieme dei permessi relativi al contesto di riferimento.

I requisiti che devono essere soddisfatti dai *contesti* sono i seguenti:

- *Identificativo univoco* (ogni *contesto* è caratterizzato da un ID all'interno del sistema).
- *Struttura gerarchica* (un *contesto* può essere contenuto all'interno di un altro *contesto* e può contenere a sua volta altri *contesti*).
- *Tutti gli oggetti devono avere un ID di contesto!* (tale ID deve riferire un contesto esistente o deve avere valore NULL. Il significato del NULL può dipendere dall'implementazione. Ad esempio a NULL può essere fatto corrispondere un *contesto* globale costituito dal sistema stesso).

Il modello ad oggetti di *WEB@WORK*, infine, include la possibilità di definire una nozione di relazione tra oggetti, per esprimere asserzioni come "*l'oggetto X è legato all'oggetto Y tramite la relazione R*" associando ad esse delle proprietà.

#### **7.1.1.2 Requisiti dell'interfaccia di programmazione**

L'API del core engine deve consentire allo sviluppatore le seguenti azioni:

- *Creare nuove classi*
  - *Creare una classe*
  - *Creare una sottoclasse*
  - *Creare una "classe relazione"*
- *Modificare classi* (aggiunta ed eliminazione di proprietà con propagazione alle sottoclassi).
- *Eliminare classi* (vengono eliminate tutte le istanze della classe con la possibilità di propagare a tutte le istanze delle sottoclassi relative e alle sottoclassi stesse).
- *Creare istanze di oggetti* (oltre a poter specificare i valori delle proprietà deve essere possibile specificare il contesto di validità dell'istanza).
- *Eliminare istanze di oggetti* (un oggetto può essere eliminato solo se non è referenziato da altri oggetti, opzionalmente è definibile un meccanismo di propagazione agli oggetti correlati).
- *Creare ed eliminare relazioni*
- *Creare ed eliminare contesti*
- *Impostare i valori delle proprietà per un oggetto*
- *Recuperare i valori delle proprietà per un oggetto*

## 7.1.2 Disegno del sistema

Di seguito viene descritto il disegno del modello ad oggetti di *WEB@WORK* sia per quanto riguarda le strutture di dati che l'interfaccia di programmazione.

### 7.1.2.1 Disegno del modello dei dati

Il modello dei dati del *core engine* di *WEB@WORK*, così come quello dei moduli, si articola su due livelli:

- Livello dei metadati
- Livello dati operazionali

Il livello dei dati operazionali dipende da quello dei metadati, che quindi verrà discusso per primo. Nella descrizione che segue, si riportano delle versioni abbreviate delle definizioni di diverse tabelle, per riflettere informazioni di progetto. Le versioni estese utilizzate per l'implementazione fisica sono riportate nel documento di analisi funzionale del Core Engine "**ECW-CE-AF-05-2001.DOC**".

#### 7.1.2.1.1 Metadati

Il livello dei metadati per gli oggetti di *WEB@WORK* è incentrato su tre tabelle principali che memorizzano rispettivamente le classi di oggetti, le proprietà e le classi relazione e su ulteriori tre tabelle: la prima è la tabella dei linguaggi, e le altre due sono tabelle che contengono i metadati "localizzati", dipendenti cioè dalla lingua. Caratteristica essenziale del sistema è infatti quella di fornire un supporto nativo alla multilinguallità.

La tabella che implementa le classi di oggetti è la seguente:

```
create table ecw_obj_class (
    obj_class          varchar2(128) primary key,
    superclass         references ecw_obj_classes (obj_class)
    is_abstract        varchar2(1) default 'n' not null,
    table_name         varchar2(30) not null unique,
    id_column          varchar2(30) not null,
    name_function      varchar2(30),
    class_ext_table    varchar2(30)
)
```

Ogni riga della tabella corrisponde ad una classe di oggetti all'interno di *WEB@WORK*. Se la classe di oggetti è una sottoclasse il campo "*superclass*" riporta l'ID della superclasse relativa. Viene supportata la nozione di "*classe astratta*", definibile come classe che non contiene istanze. Classi di questo genere esistono solo in quanto esistono delle sottoclassi derivate. Un esempio potrebbe essere costituito da una classe "*forma*" che contiene caratteristiche comuni a tutte le forme, ma che venga utilizzata solo per derivare sottoclassi che rappresentino forme reali e concrete, come cerchi, quadrati, triangoli, etc.

Ciascuna classe di oggetti definisce una tabella di database dove memorizzare le istanze degli oggetti della classe, deve essere definito anche il nome della colonna che costituisce la chiave primaria in questa tabella (dove memorizzato l'identificatore ID dell'oggetto). C'è poi anche la possibilità di definire una ulteriore tabella per la memorizzazione di ulteriori proprietà generiche.

La tabella che implementa le proprietà delle classi di oggetti è la seguente:

```
create table ecw_properties (
    id_property        number primary key,
    obj_class          not null references ecw_obj_classes
                      (obj_class),
```

```

property_name      varchar2(128) not null,
sort_order         number not null,
data_type          varchar2(30) not null,
default_val        varchar2(4000),
storage_type       varchar2(8) default 'specific' check
                  (storage_type in('specific','generic')),
min_cardinality    number default 1 not null,
max_cardinality    number default 1 not null,
is_static          varchar2(1) default 'n' not null
    )

```

Le cose da notare sono:

- Ogni proprietà ha un identificatore univoco ed è necessariamente associata ad una classe di oggetti.
- La colonna "datatype" non si riferisce al tipo di dato SQL della proprietà. Si tratta piuttosto di un nome logico ("String", "Number", etc.) del tipo di dato.
- La colonna "sort\_order" contiene l'informazione sull'ordinamento della proprietà nell'ambito di una classe di oggetti.
- Il campo "storage\_type" determina come verrà gestita la memorizzazione fisica dei valori delle proprietà. Scegliere "specific", che rappresenta il default, significa rappresentare la proprietà con una colonna della tabella indicata dal valore del campo "table\_name" della tabella "ecw\_obj\_classes". Scegliere "generic" significa utilizzare un'unica tabella per i valori di tutti gli attributi (come accade attualmente in Archdoc), nella quale ad ogni valore possibile corrisponde un record. La prima soluzione è più funzionale e performante, la seconda più semplice da gestire, ma limitata (l'aggiunta di proprietà non comporta l'aggiunta di colonne alla tabella relativa, ma semplicemente di record).
- Le colonne "min\_cardinality" e "max\_cardinality" specificano il numero di valori di una proprietà.
- La colonna "is\_static" indica se il valore dell'attributo debba essere condiviso da tutte le istanze della classe.

La tabella che implementa le proprietà delle classi di relazione è la seguente:

```

create table ecw_rel_classes (
    rel_class          varchar2(128) not null references
                    ecw_obj_classes(obj_class),
    obj_class_first    not null references ecw_obj_classes
                    (obj_class),
    role_first         references ecw_rel_roles_loc(role),
    obj_class_second   not null references ecw_obj_classes
                    (obj_class),
    role_second        references ecw_rel_roles_loc(role),
    min_card_rels_first number default 0 not null,
    max_card_rels_first number,
    min_card_rels_second number default 0 not null,
    max_card_rels_first number,
    )

```

Le cose da notare sono:

- La relazione è tra oggetti delle classi “obj\_class\_first” e “obj\_class\_second”. Quindi, ogni istanza di questa classe di relazione sarà costituita da una coppia di oggetti delle classi relative.
- Le colonne “role\_first” e “role\_second” indicano delle sigle per i ruoli giocati dagli oggetti delle classi che partecipano alla relazione.
- Le ultime quattro colonne consentono allo sviluppatore di definire dei vincoli rispetto a quanti oggetti di una delle classi siano relati a quanti dell'altra classe.

La tabella che implementa i linguaggi è la seguente:

```
create table ecw_languages (
    id_language          number primary key,
    language             varchar2(64) not null
)
```

Infine, le tabelle che implementano i dati localizzati sono:

```
create table ecw_obj_classes_loc (
    obj_class           not null references
                        ecw_obj_classes(obj_class),
    id_language         not null references
                        ecw_languages(id_language),
    label_name          varchar2(128) not null,
    label_plural        varchar2(128) not null
)
```

```
create table ecw_obj_properties_loc (
    id_property         not null references
                        ecw_obj_properties(id_property),
    id_language         not null references
                        ecw_languages(id_language),
    label_name          varchar2(128) not null,
    label_plural        varchar2(128) not null
)
```

Per tirare le somme, le tabelle “ecw\_obj\_classes” e “ecw\_obj\_properties” memorizzano i metadati che descrivono qualsiasi classe di oggetti nel sistema. La tabella “ecw\_rel\_class” memorizza le informazioni relative alle classi di relazione. Questa parte del modello dei dati è qualcosa di analogo a quello che il dizionario dei dati rappresenta in un DBMS. Le informazioni memorizzate a questo livello servono a definire i dati memorizzati al livello “operazionale”, che verranno discussi nel paragrafo successivo.

#### 7.1.2.1.2 Dati Operazionali

La tabella principale è quella di seguito riportata:

```
create table ecw_objects (
    id_object           number primary key,
    obj_class           not null references ecw_obj_classes
                        (obj_class),
    id_scope            references ecw_object(id_object),
    security_from_context
                        varchar2(1) default 'y' not null
                        check (security_from_context
                        in('y','n')),
)
```



```

creation_user      number,
creation_date      date default sysdate not null,
creation_ip        varchar2(32),
modified           date default sysdate not null,
modifyng_user      number,
modifyng_ip        varchar2(32)
)

```

Come precisato in precedenza, i contesti di sicurezza sono organizzati gerarchicamente e sono modellati come oggetti. La gerarchia dei contesti viene memorizzata in una opportuna tabella per la quale si rimanda all'appendice 2.

Altre tabelle del "core engine" memorizzano informazioni aggiuntive relate agli oggetti. Vi sono ad esempio le tabelle "ecw\_property\_values" e "ecw\_static\_prop\_values", utilizzate valori secondo lo schema di storage definito come "generic" all'interno della tabella "ecw\_obj\_classes". Per un riferimento completo alla struttura dei dati del "core engine" si rimanda alla consultazione dell'Appendice 2.

### 7.1.2.2 Disegno dell'interfaccia di programmazione

#### 7.1.2.2.1 Livello Metadati

Il modello ad oggetti di *WEB@WORK* fornisce una *API* per creare nuove classi di oggetti e definirne le proprietà. Due procedure, "new\_class" e "destroy class" vengono a questo scopo utilizzate. I prototipi di queste due procedure sono riportati di seguito.

```

procedure new_class (
    obj_class      in    ecw_obj_classes.obj_class%TYPE,
    id_language    in    ecw_languages.id_language%TYPE,
    label_name     in    ecw_obj_properties_loc.label_name%TYPE,
    label_plural   in    ecw_obj_properties_loc.label_plural%TYPE,
    superclass     in    ecw_obj_classes.superclass%TYPE    default
                        'ecw_object',
    is_abstract    in    ecw_obj_classes.is_abstract%TYPE    default
                        'n',
    table_name     in    ecw_obj_classes.table_name%TYPE    default
                        NULL,
    id_column      in    ecw_obj_classes.id_column%TYPE      default
                        'XXX',
    name_function  in    ecw_obj_classes.name_function%TYPE
                        default NULL,
    class_ext_table in    ecw_obj_classes.class_ext_table%TYPE
                        default NULL
);

procedure destroy_class (
    obj_class in    ecw_obj_classes.obj_class%TYPE,
    cascade   in    varchar2(1) default 'n'
);

```

Nella seconda procedura il parametro di input "cascade" indica se debbano essere eliminate anche le sottoclassi della classe in esame.



Un 'interfaccia analoga viene definita per la gestione degli proprietà delle classi di informazione:

```
function new_property (
    obj_class      in    ecw_obj_classes.obj_class%TYPE,
    property_name  in    ecw_obj_properties.property_name%TYPE,
    id_language    in    ecw_languages.id_language%TYPE,
    label_name     in    ecw_obj_properties_loc.label_name%TYPE,
    label_plural   in    ecw_obj_properties_loc.label_plural%TYPE,
    sort_order     in    ecw_obj_properties.sort_order%TYPE,
    data_type      in    ecw_obj_properties.data_type%TYPE,
    default_val    in    ecw_obj_properties.default_val%TYPE,
    storage_type   in    ecw_obj_properties.storage_type%TYPE,
    min_cardinality in    ecw_obj_properties.min_cardinality%TYPE,
    max_cardinality in    ecw_obj_properties.max_cardinality%TYPE,
    is_static      in    ecw_obj_properties.is_static%TYPE
) return ecw_properties.id_property%TYPE;

procedure destroy_property (
    obj_class      in    ecw_obj_classes.obj_class%TYPE,
    property_name  in    ecw_obj_properties.property_name%TYPE
);
```

Sulla base di quanto descritto, è possibile fare un esempio di come integrare all'interno del modello ad oggetti di *WEB@WORK* un proprio modello di dati. I passi da seguire sono:

- Creare una tabella che memorizzi le istanze della classe di oggetti da integrare nel sistema.
- Chiamare la procedura "*new\_class*" per definire a livello di metadati il nuovo tipo di oggetti. Se si desidera che le istanze della classe debbano essere considerate come oggetti del sistema, la nuova classe deve essere una sottoclasse della classe predefinita "*ecw\_object*".
- Chiamare la funzione "*new\_property*" per definire le proprietà della classe.

Supponendo, ad esempio, di creare la classe "*note*", per rappresentare le annotazioni da allegare ad un documento, si dovrà definire la tabella relativa:

```
create table note
(
    id_nota      references ecw_objects(id_object),
    testo varchar2(512)
);
```

Poi, con PL/SQL, si aggancia la nuova classe alla tabella relativa di metadati.

```
declare
    id_prop_appo      ecw_properties.id_property%TYPE;
begin
    new_class
    (
        superclass      =>    'ecw_object',
        id_language      =>    'IT',
        obj_class        =>    'nota'
```

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 42 di 80

```

        label_name      =>    'Nota'
        label_plural    =>    'Note',
        table_name      =>    'note',
        id_column       =>    'id_nota'

    );
    id_prop_appo:=new_property
    (
        obj_class        =>    'nota',
        property_name    =>    'testo',
        data_type        =>    'string',
        id_language      =>    'IT',
        label_name       =>    'Testo',
        label_plural     =>    'Testo'
    );
    ... (altre proprietà)
    commit;
end;
```

In questo modo, con poche righe di codice analoghe a quelle precedenti le “note” risultano agganciate ai servizi generici disponibili per tutti gli oggetti del sistema. Tale codice non deve essere modificato quando vengano aggiunti nuovi servizi generici al sistema.

L'interfaccia di programmazione a livello dei metadati comprende anche il supporto per la definizione di classi di relazioni tra classi di oggetti.

Le funzioni principali sono quelle che consentono la creazione e l'eliminazione di una classe di relazioni:

```

procedure new_rel_class (
    rel_class           in ecw_rel_classes.rel_class%TYPE,
    id_language         in ecw_languages.id_language%TYPE,
    label_name          in ecw_obj_classes_loc.label_name%TYPE,
    label_plural        in ecw_obj_classes_loc.label_plural%TYPE,
    supertype           in ecw_obj_classes.superclass%TYPE
                        default 'relationship',
    table_name          in ecw_obj_classes.table_name%TYPE,
    id_column           in ecw_obj_classes.id_column%TYPE,
    is_abstract         in ecw_obj_classes.is_abstract%TYPE default 'N',
    class_ext_table     in ecw_obj_classes.type_extension_table%TYPE
                        default null,
    function_name       in in acs_obj_classes.function_name%TYPE default
null,
    obj_class_first     in ecw_rel_classes.obj_class_first%TYPE,
    role_first         in ecw_rel_classes.role_first%TYPE default null,
    min_card_rels_first in ecw_rel_classes.min_card_rels_first%TYPE,
    max_card_rels_first in ecw_rel_classes.max_card_rels_first%TYPE,
    object_class_second in ecw_rel_classes.object_class_second%TYPE,
```

```

        role_second          in ecw_rel_classes.role_second%TYPE default
null,
        min_card_rels_second in ecw_rel_classes.min_card_rels_second%TYPE,
        max_card_rels_second in ecw_rel_classes.max_card_rels_second%TYPE
    );

procedure destroy_rel_class (
    rel_class          in ecw_rel_classes.rel_class%TYPE,
    cascade            in varchar2 default 'N'
);

```

#### 7.1.2.2.2 Livello dati operazionali

L'altra importante parte dell'interfaccia di programmazione è costituita dalle procedure e funzioni che consentono la creazione e la gestione di oggetti. Questa parte dell'API è realizzata per soddisfare la necessità di creare oggetti ed effettuare query sulle loro proprietà.

Tuttavia, è difficile costruire procedure generalizzate per l'effettuazione di query su larga scala all'interno del sistema ad oggetti. Quindi gli sviluppatori dovranno probabilmente familiarizzare con il modello dei dati ad un livello più basso, effettuando le query direttamente sulle tabelle del modello di dati, incapsulando tali query in stored procedures.

Le due procedure fondamentali sono quelle che permettono la creazione e l'eliminazione di un oggetto.

```

function new_object      (
    id_object          in    ecw_objects.id_object%TYPE default NULL,
    obj_class          in    ecw_objects.obj_class%TYPE      default
                        'ecw_object',
    creation_date      in    ecw_objects.creation_date%TYPE,
    creation_user      in    ecw_objects.creation_user%TYPE,
    creation_ip        in    ecw_objects.creation_ip%TYPE,
    id_scope           in    ecw_objects.id_scope%TYPE
) return ecw_objects.id_object%TYPE;

procedure destroy_object      (
    id_object          in    ecw_objects.id_object%TYPE
);

```

Vengono poi definite alcune funzioni generiche per manipolare gli attributi. Per recuperare come gli attributi siano memorizzati e per recuperare il singolo valore di un attributo.

```

procedure get_property_storage      (
    id_object          in    ecw_objects.id_object%TYPE,
    property_name      in    ecw_properties.property_name%TYPE,
    p_column           out   varchar2,
    p_table_name       out   varchar2,
    p_key_sql          out   varchar2
);

function get_property      (

```

```

id_object      in      ecw_objects.id_object%TYPE,
property_name  in      ecw_properties.property_name%TYPE
) return varchar2;

```

```

procedure set_property (
    id_object      in      ecw_objects.id_object%TYPE,
    property_name  in      ecw_properties.property_name%TYPE,
    value_in       in      varchar2
);

```

Proseguendo l'esempio di prima dell'implementazione delle "note", si potrebbe definire, ad esempio, la seguente funzione per la creazione di una nuova nota:

```

function new_nota (
    id_nota      in      note.id_nota%TYPE,
    descrizione  in      note.descrizione%TYPE
) return note.id_nota%TYPE
is
    id_nota      _appo ecw_object.id_object%TYPE;
begin
    id_nota_appo:=ecw_objects.new_object
    (
        id_object =>    id_nota,
        obj_class =>    'nota',
        ...
    );
    insert into note (id_nota,descrizione) values
    (id_nota_appo,descrizione);
    return id_nota_appo
end new_nota;

```

Per riassumere, per trarre profitto dai servizi applicativi generalizzati di *WEB@WORK*, un nuovo modulo applicativo deve realizzare tre operazioni:

- Definire un modello di dati per descrivere gli oggetti dell'applicazione.
- Creare una classe di oggetti relativa.
- Assicurarsi che gli oggetti siano creati con la procedura "*new\_object*" del package "*ecw\_objects*" oltre ad inserire le opportune informazioni nelle tabelle del modello di dati specifico.

In tal modo i servizi generali dell'infrastruttura applicativa (permessi, etc.) potranno lavorare con tutti gli oggetti registrati nella tabella "*ecw\_objects*", quindi anche con gli oggetti del modello di dati in esame (Es. con le *note* dell'esempio precedente). In generale sono perciò necessari tre passi per *agganciare* un modulo applicativo ai servizi generali di *WEB@WORK*.

L'interfaccia di programmazione a livello dei dati operazionali comprende anche il supporto per la definizione di di relazioni tra oggetti.

Le funzioni principali sono quelle che consentono la creazione e l'eliminazione di una istanza si relazione:

```
function new_rel (  
    id_rel          in ecw_rels.id_rel%TYPE default null,  
    rel_class       in ecw_rels.rel_class%TYPE default 'relationship',  
    object_id_first in ecw_rels.object_id_first%TYPE,  
    object_id_second in ecw_rels.object_id_second%TYPE,  
    id_scope        in ecw_objects.id_scope%TYPE default null,  
    creation_user   in ecw_objects.creation_user%TYPE default null,  
    creation_ip     in ecw_objects.creation_ip%TYPE default null  
    ) return ecw_rels.id_rel%TYPE;  
  
procedure destroy_rel (  
    id_rel          in acs_rels.id_rel%TYPE  
    );
```

## 7.2 Il sistema di autorizzazioni

### 7.2.1 Visione generale

Qualsiasi sistema multiutente deve confrontarsi con il problema generale di *chi possa fare cosa*. Per i servizi applicativi basati sul web, che tipicamente coinvolgono un gran numero di utenti, la gestione dei permessi rappresenta un aspetto critico: l'accesso ai contenuti, ai servizi e alle informazioni deve essere controllato.

Il sistema di permessi di *WEB@WORK* espone un'interfaccia di programmazione consistente per applicazioni che abbiano bisogno di gestire permessi di accesso alle risorse del sistema. In questo modo si evita di scrivere più volte del codice per gestire situazioni ricorrenti dal punto di vista dei permessi.

Attualmente *Archdoc* gestisce l'accesso a funzioni applicative (Es. ricerca, inserimento, modifica) relativamente ad una tipologia di documenti, quindi per tutti i documenti della stessa classe. Dal punto di vista dei privilegi di accesso, il sistema gestisce attualmente dei ruoli, intesi come raggruppamenti di funzioni applicative da poter assegnare di volta in volta ai vari utenti del sistema.

Il sistema di permessi di *WEB@WORK* è costituito da due componenti fondamentali:

- Un'API per la gestione dei controlli di accesso nelle varie applicazioni.
- Un'interfaccia *web-based* indirizzata agli amministratori di sistema per l'assegnazione e la revoca dei permessi.

La caratteristica chiave del sistema di permessi di *WEB@WORK* è la *consistenza*, sia da un punto di vista di una comune interfaccia di amministrazione che da quella di un controllo di accesso espletato e controllato in maniera semplice.

Il problema che un sistema di controllo degli accessi deve risolvere può essere messo in paragone con le componenti grammaticali di una frase semplice, che generalmente si compone di *soggetto*, *verbo* e *complemento oggetto*. Può un dato soggetto effettuare una determinata operazione su un dato oggetto? Il soggetto è l'*attore*, al quale possono essere associati zero o più utenti; l'oggetto è ciò su cui l'azione deve essere effettuata, ci si riferisce all'oggetto o agli oggetti anche con il termine *target*. Il verbo è l'operazione che deve essere realizzata. Se un'operazione si chiama "*pippo*" ci si può riferire a volte al privilegio "*pippo*", con ciò volendo dire che l'utente ha il privilegio di eseguire quella operazione.

## 7.2.2 Analisi dei requisiti

I requisiti che il sistema di permessi di *WEB@WORK* rispetta possono essere ripartiti nelle seguenti categorie:

- **Requisiti funzionali**

- *Granularità*: il sistema deve consentire di governare il controllo di accesso a livello di singolo oggetto, cioè a livello di singola riga nel modello ad oggetti.
- *Operazioni*:
  - Controllo di base (Può il l'attore A effettuare l'operazione O sul target T?)
  - Verifica sugli *attori* (Quali attori A possono effettuare l'operazione O sul target T?)
  - Verifica sulle operazioni (Quali operazioni O possono essere effettuate sul target T dall'attore A?)
  - Verifica sui *target* (Su quali target l'attore A può effettuare l'operazione O?)

- **Requisiti applicativi**

- *Aggregazione di privilegi* (Può essere comodo raggruppare i privilegi, ad esempio il privilegio "admin" su un ipotetico target comporta automaticamente i privilegi di "read", "write" e "delete" su di esso).
- *Aggregazione di attori*
- *Ambito del controllo di accesso*
  - *Scope* (Il privilegio di accesso deve essere, se non specificato, ereditato dal contesto in esame. Ad esempio, se non si ha accesso ad un forum di discussione automaticamente non si hanno i privilegi di lettura per i messaggi del forum).
  - *Override* positivo (E' la possibilità di fornire dei privilegi in più rispetto a quelli ereditati dal contesto).
  - *Override* negativo (E' la possibilità di eliminare un sottoinsieme di privilegi da quelli ereditati dal contesto).
- *Efficienza*
- *Facilità d'uso*

### 7.2.2.1 Il modello dei dati

Per una descrizione dettagliata del modello dei dati del sistema di controllo degli accessi di *WEB@WORK* si rimanda all'Appendice 2. Esso è comunque relativamente semplice e consta di cinque tabelle principali:

- *ecw\_methods* → L'insieme di tutte le operazioni dell'applicazione
- *ecw\_privileges* → L'insieme di tutti i privilegi
- *ecw\_privilege\_method\_rules* → Associativa tra operazioni e privilegi
- *ecw\_privilege\_tree* → Associazione di struttura tra privilegi
- *ecw\_permissions* → Una tabella con una riga per ogni privilegio direttamente assegnato su qualsiasi oggetto del sistema.

Oltre alle precedenti, vi sono una serie di viste che hanno lo scopo dare risposta a questioni specifiche circa i permessi. Ad esempio, le tabelle precedenti descrivono permessi diretti o espliciti. La presenza dei meccanismi di ereditarietà e dei valori di default possono però introdurre dei permessi che non sono esplicitamente definiti (ad esempio l'accesso in lettura ad un forum di discussione determina l'accesso in lettura a tutti i messaggi del medesimo).

### 7.2.2.2 L'interfaccia di programmazione

L'interfaccia di programmazione del sistema di permessi è costituita da due semplici procedure che consentono di assegnare o revocare i privilegi:

```
procedure grant_permission (
    id_object      ecw_permissions.id_object%TYPE,
    id_grantee     ecw_permissions.id_grantee%TYPE,
    privilege      ecw_permissions.privilege%TYPE
);

procedure revoke_permission (
    id_object      ecw_permissions.id_object%TYPE,
    id_grantee     ecw_permissions.id_grantee%TYPE,
    privilege      ecw_permissions.privilege%TYPE
);
```

## 7.3 Utenti e gruppi

### 7.3.1 Visione generale

Un servizio web based che debba venire incontro alle necessità di una grande organizzazione deve essere in grado di modellare le strutture organizzative attraverso varie modalità di decomposizione delle strutture stesse. Ad esempio una organizzazione può essere pensata suddivisa in strutture di competenza o in strutture definite in base al territorio. *Arcdoc2001* mette a disposizione un pieno supporto a questo genere di situazioni utilizzando diversi concetti, descritti in seguito.

Per una panoramica generale il punto di partenza può essere rappresentato dal concetto di *gruppo*. Un *gruppo* è composto da membri che, a loro volta possono essere costituiti da altri gruppi. Oltre alla relazione di appartenenza viene definita una relazione di *composizione* tra gruppi. Un *gruppo* può essere composto di altri gruppi (gruppi componenti).

Un *gruppo Gc* può essere membro e/o componente di un altro gruppo *Gp*. La differenza consiste nel modo in cui i membri di *Gc* sono in relazione con *Gp*.

- Se l'attore A è membro (componente) di *Gc* e se *Gc* è *componente* di *Gp*, allora A è membro (componente) di *Gp*.
- Se l'attore A è membro (componente) di *Gc* e *Gc* è *membro* di *Gp*, allora non c'è alcuna relazione tra A e *Gp* come conseguenza della relazione tra *Gc* e *Gp*.

Volendo fare un esempio, tanto per fissare le idee, si consideri Engiweb.com come appartenente ad Engineering Ingegneria Informatica. Engiweb.com ha diverse laboratori di sviluppo, tra cui quello di Roma. Se Marco Caressa è membro del laboratorio Roma, egli è automaticamente membro di Engiweb.com. Tuttavia ciò non fa di lui un membro di Engineering Ingegneria Informatica.

In *WEB@WORK* Engineering Ingegneria Informatica, Engiweb.com e i laboratori di sviluppo di questa sarebbero modellati come *gruppi*, e Marco Caressa come un *utente*. Ci sarebbe una relazione di composizione tra i laboratori di sviluppo ed Engiweb.com. Ci sarebbe una relazione di appartenenza tra Engiweb.com ed Engineering Ingegneria Informatica. Ciò non comporta però alcuna relazione automatica tra Engineering Ingegneria Informatica e Marco Caressa.



### 7.3.2 Analisi dei requisiti

#### 7.3.2.1 Il modello dei dati

Il modello dei dati a supporto della rappresentazione degli *attori* e dei *gruppi* del sistema deve consentire le seguenti tipologie di entità:

- *Attori*

Un *attore* è una entità utilizzata per rappresentare sia un *gruppo* che una *persona*.

Il modello dei dati deve tener conto dei seguenti vincoli:

- Un *attore* ha un indirizzo di e-mail, che può anche essere vuoto.
- Un *attore* può avere più indirizzi di e-mail associati ad esso.
- L'indirizzo di e-mail di un *attore* deve essere unico all'interno del sistema.

- *Gruppi*

Un *gruppo* è una collezione di zero o più *attori*.

Il modello dei dati deve poter consentire il *subclassing* di un gruppo come *oggetto* del sistema:

- *Persone*

Una *persona* rappresenta un *essere umano*, passato o presente del sistema.

- Una *persona* deve avere associato un nome.

- *Utenti*

Un *utente* è una *persona* registrata nel sistema. Un *utente* può avere attributi aggiuntivi, come un *nickname*.

Il modello dei dati deve tener conto dei seguenti vincoli:

- Un *utente* deve avere un indirizzo di e-mail non vuoto.
- Due *utenti* non possono avere lo stesso indirizzo di e-mail nell'ambito di una installazione del sistema (quindi l'indirizzo di e-mail identifica un singolo utente all'interno del sistema).
- Un *utente* può avere più indirizzi di e-mail (due o più indirizzi di e-mail identificano un singolo utente).
- Un *utente* è caratterizzato da una *password*.

Il modello dei dati per *attori* e *gruppi* deve fornire supporto per le seguenti tipologie di relazione tra le entità:

- *Appartenenza*

Un *attore* A è considerato membro del gruppo G

- Quando esiste una relazione diretta di appartenenza tra A e G
- Quando esiste una relazione diretta di appartenenza tra A e un qualsiasi gruppo Gc e Gc è *componente* di G

Un *attore* può essere membro di più gruppi.

Un *attore* può essere membro dello stesso gruppo più volte solo per relazioni di appartenenza di tipo differente. Ad esempio Marco può appartenere ad Engiweb sia come dipendente che come quadro.

Non è supportato il fatto che un *attore* possa appartenere a se stesso.

- *Composizione*

Un *gruppo* Gc è considerato componente di un secondo gruppo Gp

- Quando esiste una relazione diretta di composizione tra Gc e Gp

- Quando esiste una relazione diretta di composizione tra Gc e un qualsiasi gruppo Gi e Gi è *componente* di Gp
- Un *gruppo* può essere componente di più gruppi.
- Non è supportato il fatto che un *gruppo* possa essere componente di se stesso.

### 7.3.2.2 L'interfaccia di programmazione

L'API fornisce agli sviluppatori le primitive per la realizzazione delle seguenti operazioni:

- Creazione di un gruppo.
- Creazione di una persona.
- Creazione di un utente.
- Promozione di una persona ad utente.
- Regressione di un utente a persona.
- Modifica di un attore: (possibilità di aggiungere, modificare ed eliminare attributi per un attore).
- Recupero degli attributi di un attore.
- Eliminazione di un attore:
  - Possibilità di rimuovere l'attore da tutti i gruppi, e quindi eliminare l'attore con unica chiamata API.
  - Nel caso di un gruppo, possibilità di rimuovere tutti gli attori da un gruppo e quindi eliminare il gruppo stesso con una singola chiamata API.
- Aggiunta/rimozione di un attore come membro di un gruppo.
- Aggiunta/rimozione di un gruppo come componente di un altro gruppo.
- Rimozione di un attore.
- Controllo di appartenenza: possibilità di rispondere con una chiamata API alla seguente domanda: "E' l'attore A membro del gruppo G?"
- Controllo di composizione: possibilità di rispondere con una chiamata API alla seguente domanda: "E' il gruppo Gc componente del gruppo Gp?"
- Query di recupero dei membri: possibilità di rispondere con una chiamata API alla seguente domanda: "Quali attori sono membri del gruppo G?"
- Query di recupero dei componenti: possibilità di rispondere con una chiamata API alla seguente domanda: "Quali gruppi sono componenti del gruppo G?"
- Query di appartenenza: possibilità di rispondere con una chiamata API alla seguente domanda: "Di quali gruppi l'attore A è membro?"
- Query di composizione: possibilità di rispondere con una chiamata API alla seguente domanda: "Di quali gruppi il gruppo G è componente?"
- Controllo della possibilità di appartenenza: possibilità di rispondere con una chiamata API alla seguente domanda: "L'attore A può divenire membro del gruppo G?"
- Controllo della possibilità di composizione: possibilità di rispondere con una chiamata API alla seguente domanda: "Il gruppo Gc può diventare componente del gruppo Gp?"

E' importante rimarcare due requisiti fondamentali nell'espletamento di tali servizi applicativi. Il primo è quello dell'efficienza. La maggior parte delle pagine di un sito web debbono controllare l'appartenenza di un utente ad un gruppo prima di visualizzare la pagina stessa o alcuni dei suoi contenuti. Il modello dei dati deve pertanto essere in grado di soddisfare una memorizzazione ed un recupero efficienti degli attributi degli *attori* in gioco e della loro appartenenza ai gruppi. Il secondo requisito è costituito dalla facilità d'uso. Poiché molte query SQL effettueranno il controllo dell'appartenenza ad un gruppo con un ulteriore predicato all'interno della relativa

clausola WHERE, quale che sia il meccanismo utilizzato per il controllo di appartenenza in SQL, questo deve essere semplice.

### 7.3.3 Disegno del sistema

Il *nucleo* del modello dei dati che implementa *gruppi* e *attori* è relativamente semplice. Tuttavia esso consente di *modellare* organizzazioni reali anche molto complesse.

Il vincolo è comunque di poter modellare gruppi usando un grafo diretto aciclico. In ogni caso, poiché le query direttamente su queste strutture potrebbero essere poco performanti, e dato che praticamente ogni pagina di un sito deve effettuare il controllo di *membership*, sarà necessario creare un insieme di viste, check constraints e tabelle ausiliarie per mantenere le prestazioni ad un livello di efficienza.

#### 7.3.3.1 Il modello dei dati

Il modello dei dati consiste delle seguenti tabelle:

##### *Attori*

L'insieme di tutti gli attori: ogni persona, utente o gruppo deve avere una riga corrispondente all'interno di questa tabella.

##### *Persone*

L'insieme di tutti le persone definite.

##### *Utenti*

L'insieme di tutti gli utenti registrati.

##### *Preferenze Utenti*

L'insieme delle preferenze di tutti gli utenti.

##### *Gruppi*

L'insieme di tutti i gruppi definiti.

##### *Tipi Gruppo*

L'insieme di tutti i tipi di gruppo definiti.

##### *Relazioni di appartenenza*

L'insieme di tutte le relazioni dirette di appartenenza tra un *attore* e un *gruppo*.

##### *Mappatura Gruppi Membri*

Tabella di mappatura tra *Attori* e i *Gruppi* cui i primi appartenengono.

##### *Relazioni di composizione*

L'insieme di tutte le relazioni dirette di composizione tra un *gruppo* e un altro *gruppo*.

##### *Mappatura Gruppi Componenti*

Tabella di mappatura tra *Gruppi* e i *Gruppi* di cui i primi sono componenti.

I nuovi gruppi possono essere creati tramite un opportuno costruttore. Vincoli di appartenenza possono essere specificati all'atto della creazione, fissando il gruppo *padre* nella chiamata al costruttore.

Le tabelle relative alle relazioni di appartenenza e di composizione indicano i membri e i componenti diretti di un gruppo. Le pagine web effettueranno spesso delle query per ricavare i vincoli di appartenenza e composizione, allo scopo di controllare i permessi di accesso.

Tuttavia, il grafo rappresentativo di tutta la struttura dei gruppi può divenire imponente e attraversarlo ogni volta che debba essere fatta una query di questo tipo può determinare un decadimento delle prestazioni applicative. Come anticipato in precedenza, vengono quindi creati

dei trigger che “vigilano” rispetto a modifiche delle relazioni di appartenenza e di composizione e aggiornano conseguentemente le tabelle di mappatura.

### 7.3.3.2 L'interfaccia di programmazione

#### Persone

Il metodo “new\_person” crea una nuova persona e restituisce l'identificativo di essa. Le uniche informazioni obbligatorie da fornire sono il nome e cognome, oltre al tipo di oggetto (obj\_class) che di default è posto a “person” e “creation\_date” che di default è posto a sysdate. L'interfaccia per questa funzione è:

```
function new_person(
    person_id          persons.person_id%TYPE,
    obj_class          ecw_objects.obj_class%TYPE,
    creation_date       ecw_objects.creation_date%TYPE,
    creation_user       ecw_objects.creation_user%TYPE,
    creation_ip         ecw_objects.creation_ip%TYPE,
    email              actors.email%TYPE,
    url                 actors.url%TYPE,
    first_names         persons.first_names%TYPE,
    last_name           persons.last_name%TYPE
) return persons.person_id%TYPE;
```

Il metodo “destroy\_person” elimina la persona identificata tramite ID. L'interfaccia per questa procedura è:

```
procedure destroy_person(
    person_id          persons.person_id%TYPE
);
```

Il metodo “name\_person” restituisce il nome della persona identificata tramite ID. L'interfaccia per questa procedura è:

```
function name_person(
    person_id          persons.person_id%TYPE
) return varchar2;
```

#### Utenti

Il metodo “new\_user” crea un nuovo utente e restituisce l'identificativo di esso. Le uniche informazioni obbligatorie da fornire sono la e-mail, il nome e cognome, oltre al tipo di oggetto (obj\_class) che di default è posto a “person” e “creation\_date” che di default è posto a sysdate. L'interfaccia per questa funzione è:

```
function new_user(
    user_id            ecw_users.user_id%TYPE,
    obj_class          ecw_objects.obj_class%TYPE,
    creation_date       ecw_objects.creation_date%TYPE,
    creation_user       ecw_objects.creation_user%TYPE,
    creation_ip         ecw_objects.creation_ip%TYPE,
    email              actors.email%TYPE,
```

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 52 di 80

```

url          actors.url%TYPE,
first_names  persons.first_names%TYPE,
last_name    persons.last_name%TYPE,
password     ecw_users.password%TYPE,
password_question ecw_users.password_question%TYPE,
password_answer ecw_users.password_answer%TYPE,
screen_name  ecw_users.screen_name%TYPE,
is_email_verified ecw_users.is_email_verified%TYPE
) return ecw_users.user_id%TYPE;

```

Il metodo “destroy\_user” elimina l’utente identificato tramite ID. L’interfaccia per questa procedura è:

```

procedure destroy_user(
    user_id      ecw_users.user_id%TYPE
);

```

Il metodo “receives\_alert” restituisce ‘t’ (VERO) se l’utente dovrebbe ricevere notifiche via e-mail, ‘f’ in caso contrario. L’interfaccia per questa funzione è:

```

function receives_alert(
    user_id      ecw_users.user_id%TYPE
);

```

I metodi “approve\_email” e “unapprove\_email” vengono utilizzati per stabilire se l’indirizzo email dell’utente è valido. Le interfacce per queste procedure sono:

```

procedure approve_email(
    user_id      ecw_users.user_id%TYPE
);

procedure unapprove_email(
    user_id      ecw_users.user_id%TYPE
);

```

## Gruppi

Il metodo “new\_group” crea un nuovo gruppo e restituisce l’identificativo di esso. Tutti i campi sono opzionali e posti di default a null, tranne il tipo di oggetto (obj\_class) che di default è posto a “group”, “creation\_date” e “group\_name” che di default è posto a sysdate. L’interfaccia per questa funzione è:

```

function new_group(
    group          ecw_groups.group_id%TYPE,
    obj_class      ecw_objects.obj_class%TYPE,
    creation_date  ecw_objects.creation_date%TYPE,
    creation_user  ecw_objects.creation_user%TYPE,
    creation_ip    ecw_objects.creation_ip%TYPE,
    email          actors.email%TYPE,
    url            actors.url%TYPE,

```

```

group_name      ecw_groups.group_name%TYPE
) return ecw_groups.group_id%TYPE;

```

Il metodo “name\_group” restituisce il nome del gruppo identificato tramite ID. L’interfaccia per questa procedura è:

```

function name_group(
group_id          ecw_groups.group_id%TYPE
) return varchar2;

```

Il metodo “is\_member” restituisce ‘t’ (VERO) se l’attore specificato è membro del gruppo specificato, altrimenti ritorna ‘f’. L’interfaccia per questa procedura è:

```

function is_member(
group_id          ecw_groups.group_id%TYPE,
actor_id          ecw_actors.actor_id%TYPE
) return varchar2;

```

## Relazioni di appartenenza

Il metodo “new\_membership\_rel” crea un nuovo tipo di relazione di appartenenza tra due attori e restituisce l’identificativo di esso. Tutti i campi sono opzionali e posti di default a null, tranne il tipo di relazione (rel\_class) che di default è posto a “membership\_rel”. L’interfaccia per questa funzione è:

```

function new_membership_rel(
rel_id            ecw_membership_rels.rel_id%TYPE,
rel_class         ecw_rel_classes.rel_class%TYPE,
object_id_first   ecw_rels.object_id_first%TYPE,
object_id_second   ecw_rels.object_id_second%TYPE,
member_state      ecw_membership_rels.member_state%TYPE,
creation_user      ecw_objects.creation_user%TYPE,
creation_ip        ecw_objects.creation_ip%TYPE,
) return ecw_membership_rels.rel_id%TYPE;

```

Il metodo “destroy\_membership\_rel” elimina la relazione di appartenenza identificata tramite ID. L’interfaccia per questa procedura è:

```

procedure destroy_membership_rel(
rel_id            ecw_membership_rel.rel_id%TYPE
);

```

## Relazioni di composizione

Il metodo “new\_composition\_rel” crea un nuovo tipo di relazione di composizione e restituisce l’identificativo di esso. Tutti i campi sono opzionali e posti di default a null, tranne il tipo di relazione (rel\_class) che di default è posto a “composition\_rel”. L’interfaccia per questa funzione è:

```

function new_composition_rel(
rel_id            ecw_composition_rels.rel_id%TYPE,
rel_class         ecw_rel_classes.rel_class%TYPE,

```

```

object_id_first      ecw_rels.object_id_first%TYPE,
object_id_second     ecw_rels.object_id_second%TYPE,
creation_user        ecw_objects.creation_user%TYPE,
creation_ip          ecw_objects.creation_ip%TYPE,
) return ecw_composition_rels.rel_id%TYPE;

```

Il metodo “destroy\_composition\_rel” elimina la relazione di appartenenza identificata tramite ID. L'interfaccia per questa procedura è:

```

procedure destroy_composition_rel(
    rel_id          ecw_composition_rel.rel_id%TYPE
);

```

## 7.4 Gestione della sicurezza

### 7.4.1 Visione generale

Virtualmente ogni sito web supporta la personalizzazione dei contenuti basata sull'identità degli utenti. Il livello di personalizzazione può essere semplice, come visualizzare il nome dell'utente sulla pagina, o più sofisticato, come la segnalazione dinamica di sezioni del sito raccomandate all'utente sulla base dei suoi comportamenti precedenti di visita al sito. In ogni caso l'identità dell'utente deve essere convalidata e resa disponibile al resto del sistema. Inoltre, alcuni siti che gestiscono servizi particolari come quelli di pagamento hanno la necessità di identificare l'utente in maniera “sicura”.

Il sistema di sicurezza consiste in realtà di una serie di sottosistemi:

#### Cookies firmati

I cookie giocano un ruolo chiave nell'immagazzinare informazioni circa il visitatore. Ad ogni modo, poiché essi sono rappresentati come *testo piano* nel computer dell'utente, la validità del cookie è un elemento importante nel considerare le informazioni in esso contenute. Allora si necessita di meccanismi per la validazione dei cookie, ma possibilmente senza accessi al database:

- Rilevamento dei tentativi di manomissione: ogni tentativo di manomissione dei cookie deve essere facilmente rilevabile dal web server.
- Prestazioni e scalabilità: l'operazione di convalida del cookie deve essere facilmente *scalabile* e non dovrebbe richiedere una query di database per ciascuna lettura del cookie.

#### Proprietà della sessione

Le applicazioni dovrebbero essere in grado di memorizzare proprietà e variabili a livello di sessione all'interno di una tabella di database.

- API: I dati a livello di sessione dovrebbero essere accessibili tramite una API.
- Pulizia: Deve essere disponibile un efficace meccanismo di pulizia per i valori di sessioni ormai scadute.

#### Login

Il sistema di sicurezza deve supportare il concetto di *login* persistente. Tale persistenza assume varie forme:

- Login permanente: Gli utenti devono essere in grado di mantenere la loro login al sistema in modo permanente senza necessità di reimmettere la password.
- Login di sessione: Il sistema di sicurezza deve supportare il concetto di sessione, con dei *token* di autenticazione che perdano validità dopo un certo periodo di tempo.
- Definizione di sessione: Una sessione è una sequenza di *click* di un utente tramite il browser nella quale essi siano separati al più da qualche costante (*timeout* di sessione).



- **Stateless:** Il sistema di sicurezza non deve prevedere il mantenimento dello stato a livello del server. Esso deve risiedere esclusivamente nelle richieste dell'utente (cookies compresi) e nel database. In questo modo l'utente potrebbe essere ridirezionato verso un differente *application server* (ad esempio per motivi di *load balancing*) senza che questo debba influire sullo stato della sua sessione.
- **Sicurezza:** Il sistema di sicurezza non dovrebbe memorizzare le password in “chiaro” all'interno del database.
- **SSL:** Il sistema deve operare quando la crittografia SSL del canale di comunicazione è realizzata al di fuori del web server (in hardware specializzato, in un firewall, etc.).

## 7.4.2 Disegno del sistema

Il sistema di sicurezza dovrebbe essere in grado di autenticare gli utenti in ambiente *sicuro* e non. Inoltre deve fornire il supporto alle sessioni al di sopra del protocollo HTTP. Le proprietà a livello di sessione vengono fornite dal sistema come servizio generico al resto di WEB@WORK.

I mattoni fondamentali utilizzati da WEB@WORK sono i seguenti:

- **Cookies:** forniscono lo stato lato client. Vengono utilizzati per l'identificazione degli utenti. La data di scadenza dei cookies viene utilizzata per terminare la sessione.
- **SHA (Secure Hash Algorithm):** consente di *firmare* i cookies garantendo che non siano stati manomessi. Viene utilizzato anche per la codifica delle password degli utenti.
- **SSL con autenticazione del server:** fornisce al client la garanzia che il server a cui si è connessi sia effettivamente il server che ci si aspetta. Fornisce inoltre il meccanismo di trasporto sicuro dell'informazione.

### 7.4.2.1 Sessioni

Utilizzando l'*expiration time* sulla firma dei cookies, è possibile determinare quando il cookie è stato inviato e quindi determinare se le due richieste fanno parte della stessa sessione.

### 7.4.2.2 Autenticazione

Sono disponibili due livelli di accesso: sicuro ed insicuro. Le informazioni per l'autenticazione sicura sono inviate solo su una connessione sicura. Ciò significa che l'utente deve ri-autenticarsi su una connessione SSL quando effettua un'attività che richieda l'autenticazione sicura.

Benché ciò riduca la facilità d'uso del sito, migliora però la sicurezza del sistema perché assicura che le informazioni di autenticazione presentate in una sezione *sicura* del sito non possano essere *sniffate*.

### 7.4.2.3 Dettagli

Il sistema di autenticazione invia sino a quattro cookies firmati, ciascuno di essi serve ad uno scopo differente. Questi cookie sono:

Nome Cookie	Valore	Durata massima	Sicuro?
ecw_session_id	session_id,user_id	Timeout di sessione	no
ecw_user_login	user_id	Indefinita	no
ecw_user_login_secure	user_id,random	Indefinita	yes

ecw_secure_token	session_id,user_id,random	Durata di sessione	yes
------------------	---------------------------	--------------------	-----

- **ecw\_session\_id**
  - Viene re-inviato ad ogni richiesta separata dalla precedente (che abbia ricevuto il cookie) da un tempo maggiore o uguale ad una determinata quantità.
  - E' valido solo per un intervallo pari al timeout di sessione
- **ecw\_user\_login**
  - Utilizzato per la login permanente
- **ecw\_user\_login\_secure**
  - Utilizzato per login permanenti sicure
  - Contiene un meccanismo di invalidazione a tempo per prevenire attacchi al sistema di *hashing*.
- **ecw\_secure\_token**
  - Dal punto di vista del browser è un cookie a livello di sessione.
  - La firma *spira* dopo un intervallo di tempo opportuno.
  - Contiene un meccanismo di invalidazione a tempo per prevenire attacchi al sistema di *hashing*.

#### 7.4.2.4 Procedimento di autenticazione

La funzione applicativa in questione viene chiamata per autenticare l'utente. Come prima cosa viene verificato il cookie *ecw\_session\_id*. Se non c'è una sessione valida in piedi ne viene creata una. Se l'utente ha un cookie di login permanente (*ecw\_user\_login* o *ecw\_user\_login\_secure*) esso viene esaminato per determinare a cosa la sessione debba essere autorizzata. Il cookie è esaminato anche per determinare se ci si trova su una connessione sicura. Se nessuno dei due cookie è presente, allora viene creata una sessione senza autenticazione.

#### 7.4.2.5 Autenticazione di connessioni sicure

Le connessioni sicure sono autenticate in maniera leggermente differente. Deve essere disponibile una funzione API che determini se la URL acceduta richieda una autenticazione sicura (è sufficiente verificare che la URL inizi con *https*).

Se è richiesta autenticazione sicura, il cookie *ecw\_secure\_token* viene verificato, per assicurarsi che i suoi dati coincidano con quelli contenuti nel cookie *ecw\_session\_id*. Ciò è vero per tutte le pagine, tranne quelle che fanno parte della procedura di login. Su tali pagine l'utente non ha ricevuto ancora il cookie *ecw\_secure\_token*, quindi non viene effettuato alcun controllo.

#### 7.4.2.6 Procedura di Login

La funzione API che verifica la login dell'utente svolge due compiti. Anzitutto manipola opportunamente i cookies di login permanenti, quindi aggiorna la sessione dell'utente per riflettere la sua *user\_id*. La manipolazione dei cookies di login permanenti si basa su tre fattori:

- Login precedente: stesso utente, altro utente.
- Persistenza: è stata richiesta una login permanente?
- Sicurezza: si tratta di una connessione sicura?

I cookies di login permanenti possono (sia per login sicura che non) sono oggetto delle seguenti tre azioni:

- Set: i cookie privi di *scadenza* sono impostati con set.

- Delete: viene effettuata la set a "" (stringa vuota) con *max age* uguale a zero, così il cookie scade immediatamente.
- Nothing: se il cookie è presente, esso rimane inalterato.

Nella tabella seguente si riportano le varie situazioni di login e le corrispondenti azioni sui cookies.

login precedente	richiesta login permanente	connessione sicura	azione su connessione sicura	azione su connessione insicura
altro utente	si	si	set	set
stesso utente	si	si	set	set
altro utente	si	no	set	delete
stesso utente	si	no	set	nothing
stesso utente	no	si	nothing	delete
altro utente	no	si	delete	delete
altro utente	no	no	delete	delete
stesso utente	no	no	delete	delete

E' presente anche una funzione di *logout*, che cancella tutti e quattro i cookies utilizzati dal sistema di autenticazione.

#### 7.4.2.7 Creazione della sessione

Le funzioni API che gestiscono le sessioni devono svolgere due compiti essenziali:

- Assicurare che vi sia un'istanza dei dati a livello di sessione per ciascun utente.
- Aggiornare la tabella degli utenti (che ha senz'altro colonne tipo *n\_sessions*, *last\_visit*, etc.)

Se non c'è una sessione già impostata ne viene creata una. Una sessione è marcata con un ID numerico generato tramite una *sequence* Oracle.

#### 7.4.2.8 Proprietà della sessione

Le variabili e le proprietà di sessione sono memorizzate in una tabella che ha l'ID di sessione in chiave. Tale tabella viene periodicamente svuotata. Per aumentare le prestazioni tale tabella può essere gestita in modalità *NOLOGGING* e la dimensione degli *extent* da aggiungere può essere resa grande per evitare frammentazione.

Il criterio di pulizia della tabella delle sessioni è quello di eliminare tutte le sessioni il cui primo *hit* sia più vecchio di un intervallo di tempo prefissato, espresso tramite un opportuno parametro di configurazione. Le proprietà di sessione vengono automaticamente eliminate con un meccanismo di *delete cascade*.

#### 7.4.2.9 Firme digitali e cookie firmati

I cookie firmati sono implementati usando il generico meccanismo di firma digitale sicura. Tale meccanismo garantisce che l'utente non possa manomettere il cookie (o "costruirlo" per suo conto) senza che ciò venga rilevato. Inoltre fornisce la possibilità opzionale di definire un *time-out* per la firma. Ciò viene realizzato semplicemente includendo una data di scadenza come parte del valore che viene firmato.

La firma viene prodotta da un'opportuna funzione e rappresenta la lista delle seguenti informazioni: *token\_id*, *expire\_time*, *hash*.

Dove ***hash* = SHA1(*value*,*token\_id*,*expire\_time*,*secret\_token*)**

Il *secret\_token* è una stringa di 40 caratteri generata randomicamente che non viene mai inviata ad alcun *user agent*. A livello dei dati si prevede un supporto tramite la seguente tabella:

```
create table secret_tokens (  
    token_id    integer constraint secret_token_pk,  
    token       char(40),  
    timestamp   sysdate  
);
```

Un'opportuna funzione di verifica della firma prende il valore del cookie e la firma e verifica che la firma sia stata generata utilizzando quel valore. Ciò può essere fatto prendendo *token\_id* ed *expire\_time* dalla firma e rigenerando il valore hash utilizzando il valore fornito e il *secret\_token* corrispondente al *token\_id*. Il valore hash rigenerato è confrontato con quello estratto dalla firma disponibile. Viene anche verificato che il tempo di scadenza sia posteriore all'istante corrente. Può essere previsto anche un *expire\_time* pari a zero, il che vuol dire che non c'è *time-out* per la firma.

### 7.4.3 API

#### 7.4.3.1 Login/Password

Le API riportate sono puramente indicative e non viene, come in precedenza, riportata la sintassi PL/SQL perché le operazioni sui cookie possono essere realizzate via PL/SQL solo nell'ipotesi di utilizzo di *Oracle Application Server* o *Oracle iAS*, dal quale comunque si vorrebbe rimanere in prospettiva svincolati.

**ecw\_user\_login** *user\_id* effettua la login dell'utente con username *user\_id*. Un flag opzionale potrebbe determinare se debbano essere emessi o meno cookie permanenti.

**ecw\_user\_logout** effettua la logout dell'utente.

**ecw\_check\_password** *user\_id password* restituisce vero o falso a seconda dei casi.

**ecw\_change\_password** *user\_id new\_password*

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 59 di 80

#### 7.4.3.2 *Firme digitali e cookie firmati*

**ecw\_sign** *value* Restituisce la firma digitale del valore *value*.

**ecw\_verify\_signature** *value signature* Restituisce vero o falso ad indicare se la firma passata come argomento coincida con quella conosciuta dal sistema.

**ecw\_set\_signed\_cookie** *cookie\_name data* Imposta un cookie firmato con il valore *data*.

**ecw\_get\_signed\_cookie** *cookie\_name* Restituisce un errore se il cookie è stato manomesso o se è stata raggiunta la data di scadenza.

#### 7.4.3.3 *Proprietà di sessione*

**ecw\_set\_client\_property** *property\_name data* imposta il valore di una proprietà di sessione.

**ecw\_get\_client\_property** *property\_name data* restituisce il valore di una proprietà di sessione.

## 8 I Moduli

### 8.1 Modulo “Workflow”

#### 8.1.1 Visione generale

Molti siti web hanno la necessità di definire processi non banali per la creazione e la manipolazione di classi di oggetti che rappresentino concetti di livello più alto, come attività da effettuare o altro. Tali processi hanno le seguenti caratteristiche:

- Agiscono nello stesso modo su tutti gli oggetti di una determinata classe.
- Per ciascun oggetto della classe, il processo è caratterizzato da una serie di transizioni, governate da regole anche complesse, che fanno “muovere” l’oggetto da uno stato iniziale verso uno o più stati finali, dopo il raggiungimento dei quali il processo si considera concluso. Con riferimento a quanto descritto a proposito del *core engine*, il processo può essere considerato come una classe in relazione con una classe di oggetti, e un’istanza del processo viene implicitamente creata ogni qualvolta venga creato un oggetto della classe correlata.

Ad, esempio, se si definisce un processo per la validazione/pubblicazione di documenti su un sito web, quando un nuovo documento viene inserito nel sistema viene creata in astratto una nuova istanza del processo di validazione/pubblicazione. Il documento parte da un o stato “inserito” e, attraverso una serie di transizioni tra stati possibili, perviene, ad esempio, ad uno stato “pubblicato” o ad uno stato “rifiutato”.

Il modulo di *wokflow* di *WEB@WORK* ha lo scopo di definire dei meccanismi astratti e riusabili per la definizione di questo tipo di processi, per coordinare le attività di più persone relativamente ad un progetto comune all’interno dei medesimi.

Alcuni esempi di workflow sono riportati di seguito:

- *Gestione di un ordine all’interno di un sito di commercio elettronico*: Il processo può comportare il pagamento tramite carta di credito, la predisposizione del bene e la sua spedizione all’acquirente.
- *Pubblicazione di contenuti*: un articolo può essere redatto dall’autore, sottoposto a verifica e validazione da parte di una persona o di un organismo preposto e infine pubblicato da una terza persona o organismo.
- *Approvazione di spese*: un dipendente compila domanda di acquisto di beni o servizi. Il suo superiore la approva, se l’ammontare dell’importo supera una certa cifra l’approvazione deve essere effettuata dal dirigente. Una volta che la spesa sia stata approvata, viene passata in amministrazione per la sua effettuazione.
- *Bug Tracking*: un utente sottomette al sistema la comunicazione di un bug di una determinata componente software. La struttura che gestisce la lista dei *bug* determina come, quando e da chi il bug debba essere risolto. La persona incaricata risolve il problema o contatta chi ha sottomesso la segnalazione per avere più informazioni. Nel caso in cui il *bug* sia risolto, l’utente può verificare che la componente software operi come ci si aspetta.

Il modulo di workflow di *WEB@WORK* fornisce un’infrastruttura centralizzata per la gestione di un’attività, caratterizzata dai seguenti macro requisiti:

- Ogni utente del sistema ha una sua “*casella personale*”, con la lista delle attività ad esso assegnate caratterizzate dalle priorità relative.
- La struttura organizzativa all’uopo preposta ha una visione generale del singolo progetto. Il modulo tiene traccia di tutte le attività del progetto (cosa sia stato fatto e da chi). L’utente che lavora al progetto può consultare tale *log* quando realizza le attività ad esso assegnate.

- La struttura manageriale ha la visibilità dell'intero processo, allo scopo di valutare in maniera sintetica il grado di efficienza del processo ed individuare eventuali *colli di bottiglia*.
- L'analista ha a sua disposizione un'interfaccia per definire e rifinire il processo di *workflow* all'interno dello specifico progetto.

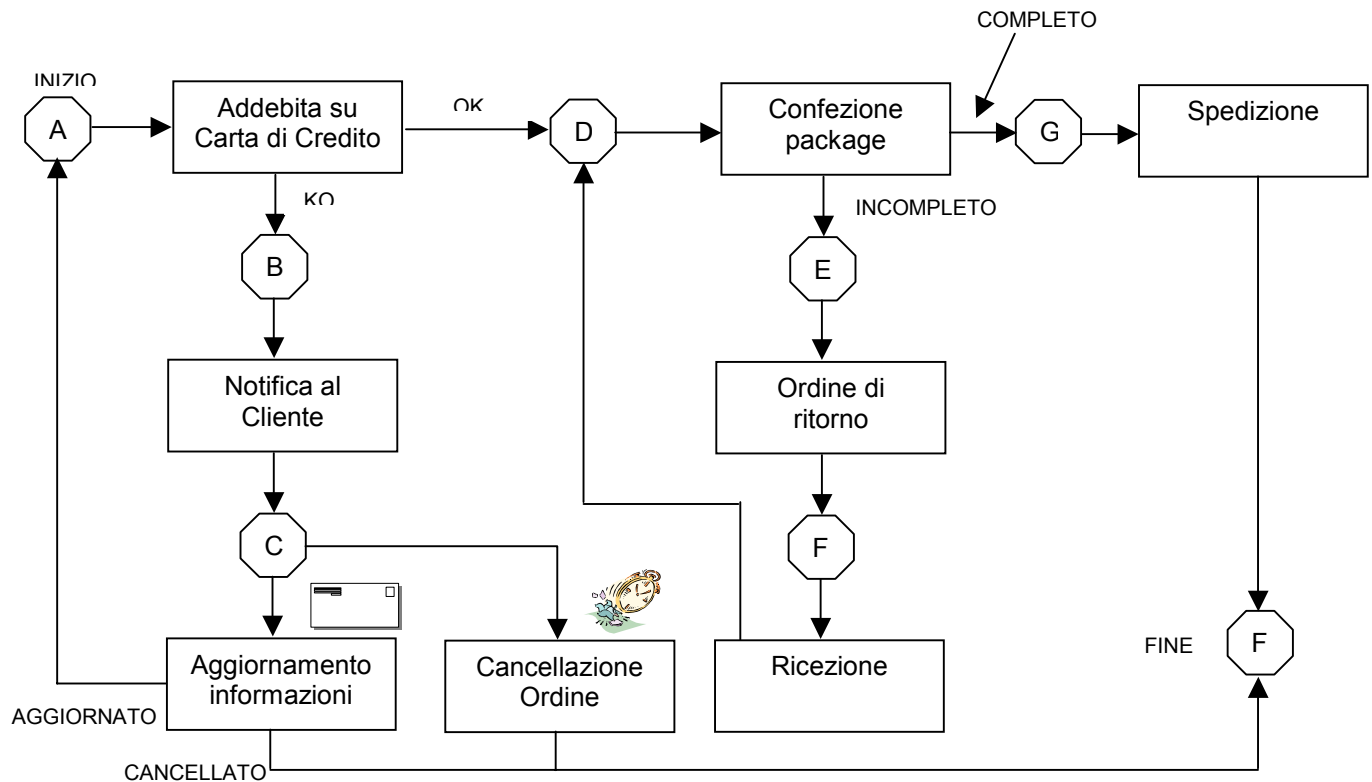
### 8.1.2 Analisi concettuale

Un workflow in *WEB@WORK* è la *definizione formale di un processo* costituito da *nodi* e *transizioni*. Rifacendosi ad una metafora relativa al generico ufficio, i *nodi* corrispondono alle "*caselle personali*" poc'anzi descritte e le transizioni alle attività che devono essere realizzate.

Nella figura seguente è indicato un esempio di ipotetico processo di acquisto di merci via Web modellato utilizzando questi elementi.

I cerchi rappresentano i nodi, i rettangoli le transizioni. I nodi sono inattivi, tutto ciò che fanno è raccogliere un insieme di informazioni (definito d'ora in avanti *token*) che rappresenti lo stato del processo. Se, ad esempio, c'è un *token* nel nodo "D" significa che si è pronti per effettuare la confezione del *package* della merce da inviare.

Le *transizioni* sono attive. Esse *muovono* i *token* dai nodi di input verso quelli di output. Quando ciò accade si dice che si *attivano*. La transizione può attivarsi solo se c'è almeno un *token* in ciascun *nodo* di input. In tal caso la transizione si dice *abilitata*. Una transizione abilitata è in grado di attivarsi.



L'istante in cui la transizione si abilita e quello in cui si attiva sono differenti. L'evento che determina l'attivazione di una transazione abilitata viene denominato *trigger*. E' possibile individuare quattro tipologie di *trigger*.



- *User trigger* - La maggior parte delle transizioni di stato sono realizzate da utenti del sistema. Il simbolo utilizzato nella figura precedente è la "freccia".
- *Message trigger* - Alcune attività sono sotto il diretto controllo del modulo di *workflow*. Il modulo applicativo riceve il messaggio che una determinata attività è stata completata. Il simbolo utilizzato nella figura precedente è la "busta".
- *Automatic trigger* - Sono le transizioni che si attivano automaticamente non appena siano abilitate. La transizione di notifica al cliente, tra i nodi B e C della figura precedente, è di questo tipo. In questo caso l'azione associata alla notifica potrebbe consistere nell'invio di una E-Mail all'utente interessato. Altre transizioni possono eseguire procedure specifiche quando vengono attivate.
- *Time trigger* - Alcune transizioni debbono essere attivate ad un preciso istante temporale. La transizione "Cancellazione Ordine" nella figura precedente ha un *time trigger*, simbolizzato da un orologio, che determina l'automatica cancellazione dell'ordine se il cliente non ha inviato le informazioni necessarie entro, ad esempio, tre settimane.

Nella figura precedente, quando un *workflow* viene attivato un *token* viene posto nello stato di "inizio". Questo abilita la transizione automatica "Carica carta di credito" (effettuazione del pagamento tramite carta di credito). Se l'operazione ha successo viene prodotto un *token* nel nodo "D", se fallisce viene prodotto nel nodo "B". La regola generale consiste nel fatto che attivare una transizione significa *consumare un token da ciascuno dei nodi di input e produrre un token in ciascun nodo di output per il quale il predicato di controllo dia esito positivo*. Il predicato di controllo è un'affermazione che può essere fatta rispetto al possibile esito della transizione che dà come risultato "vero" o "falso". La transizione "Carica carta di credito" determina esplicitamente un *instradamento* condizionale verso uno dei due rami. Tale instradamento può essere talvolta di carattere implicito, come quello che opera tra le transizioni "Aggiornamento informazioni" e "Cancellazione ordine". Poichè c'è un solo *token* nel nodo C, solo una delle due transizioni può avere luogo. Però, contrariamente all'instradamento esplicito relativo alla transizione "Carica carta di credito", dove la decisione viene presa esplicitamente non appena la transizione termina, la scelta tra "Aggiornamento informazioni" e "Cancellazione ordine" è fatta il *più tardi possibile*.

Infatti, entrambe le transizioni saranno abilitate quando c'è un *token* in C. Se l'utente aggiorna le informazioni prima che la transizione temporizzata (nell'esempio, entro le tre settimane) di cancellazione dell'ordine abbia luogo, tale transizione non avrà più luogo. E, viceversa, se, scadute le tre settimane, viene attivata la transizione di cancellazione dell'ordine, una volta che questa sia stata eseguita l'aggiornamento delle informazioni non potrà più aver luogo. Quindi in questo caso la scelta è effettuata implicitamente, basandosi su un meccanismo di temporizzazione.

Il predicato di controllo, definito in precedenza, dipende in generale dal valore di attributi che vengono impostati dalla transizione medesima (nell'esempio i valori possibili sono "OK" e "KO"). Tali valori possono avere una struttura più complessa di una semplice alternativa "vero/falso", tuttavia il risultato del predicato di controllo deve essere sempre *vero* o *falso*.

Il modulo di *workflow* gestisce anche instradamenti paralleli, dove due o più attività si verificano in modo concorrente o senza un ordine definito. Ciò viene realizzato tramite delle transizioni che producono più *token* di quanti ne consumino. Per ri-sincronizzare l'esecuzione con una transizione che si metta in attesa che tutte le transizioni concorrenti terminino prima di attivarsi essa stessa. Transizioni di questo tipo, contrariamente alle precedenti, consumano più *token* di quanti non ne producano.

Il modulo di *workflow* ha la necessità di essere integrato con codice sviluppato ad hoc per l'implementazione, ad esempio, di regole di transizione particolari. Pertanto è previsto un meccanismo di *callback* che consenta la notifica dell'esecuzione del codice personalizzato al motore di *workflow*. La modalità con cui ciò viene realizzato è quella di definire una stored procedure PL/SQL con un nome predefinito ed invocarla con un insieme di argomenti dipendente dalla situazione in esame.

Il modulo di workflow traccia tutte le operazioni durante l'esecuzione di un processo mantenendo un archivio storico costituito da un sottoinsieme del sistema di journal del *core engine*. Come specificato in precedenza, il *core engine* effettua le registrazioni (*journaling*) delle azioni effettuate su un oggetto qualsiasi del sistema.

### 8.1.3 Il modello dei dati

#### 8.1.3.1 Livello Metadati

Il modulo di workflow trae spunto, dal punto di vista del modello matematico sottostante, dalla teoria delle *Reti di petri*. Il livello dei metadati è realizzato tenendo conto di ciò e prevedendo una tabella per ciascuno degli elementi coinvolti (nodi, transizioni e archi).

Vi è poi una tabella associativa tra le transizioni e gli attributi di workflow, che specifica quale attributo debba essere considerato l'output di quale transizione.

Il concetto di *attributo di workflow* è relativo al governo dell'esecuzione del workflow medesimo. Ad esempio una delle attività di un dato workflow può consistere nell'approvare o rifiutare qualcosa. In tal caso vi sarà tipicamente un attributo *approved\_p* di tipo booleano per rappresentare un risultato che determina l'instradamento del processo verso un ramo piuttosto che un altro.

#### 8.1.3.2 Livello dati operazionali

Anche il modello dei dati operazionali segue da vicino il modello delle *Reti di Petri*. C'è una tabella, *ecw\_wf\_tasks*, che memorizza le informazioni dinamiche relative alle transizioni ed una, *ecw\_wf\_tokens* che gestisce le informazioni relative ai token (le informazioni dinamiche presenti in un nodo). Per queste tabelle, come per quelle del livello dei metadati di cui al precedente paragrafo, si rimanda al dettaglio in Appendice 3.

### 8.1.4 Interfacce (cenni)

Introdotti, sia pure in maniera non rigorosa da un punto di vista formale, i concetti fondamentali relativi al *workflow*, si può sinteticamente affermare che il modulo di workflow di *WEB@WORK* tiene traccia di quali attività debbano essere svolte all'interno di un processo, notificando via e-mail gli utenti incaricati di effettuarle ed assicurando che la medesima attività non possa venire accidentalmente eseguita da persone differenti.

Tramite esso è possibile fornire agli utenti del sistema un'interfaccia utente unificata per aiutarli a tener traccia delle attività che viene richiesto loro di effettuare. Tale interfaccia agisce come unico collettore delle informazioni relative all'attività in esame, della lista delle attività precedenti inclusi eventuali commenti e di una panoramica visuale del processo.

Viene messa a disposizione anch' un'interfaccia per analizzare le prestazioni di un processo, per individuare eventuali criticità all'interno di esso.

Il processo di workflow è sempre incentrato attorno ad un oggetto (un contenuto da pubblicare sul web, un contratto che debba seguire uno specifico iter di firma, etc.). Quindi è anzitutto necessario definire tale oggetto, creare la tabella che ne memorizzi le istanze, etc. Una volta fatto questo vi sono sostanzialmente quattro passi per trarre vantaggio dai servizi del modulo di workflow di *WEB@WORK*:

- **Modellare il processo.** La definizione del processo deve essere specificata in modo formale. La complessità di questa fase dipende evidentemente dalla complessità del processo. Il modulo mette a disposizione una semplice interfaccia web per la formalizzazione del processo.
- **Realizzare l'interfaccia utente**

- **Aggiungere un meccanismo per avviare il processo.** Normalmente si potranno utilizzare dei trigger nel Database. Ad esempio, ogni qualvolta viene creato un nuovo record nella tabella dei “*contratti*” viene generata un’istanza del processo di workflow relativo ai contratti. Ciò può essere realizzato anche a livello di *API*, in modo che la funzione che crea la nuova istanza dell’oggetto contratto avvii l’istanza del processo ad essa relativo.

Per processi di complessità non banale, si può considerare l'utilizzo di meccanismi di *callback*, messi a disposizione del modulo di workflow per consentire l'esecuzione di codice PL/SQL personalizzato. Sono presenti due meccanismi di *callback*:

- **Predicati di controllo.** Il concetto di *predicato di controllo* è stato già introdotto a proposito dell'instradamento condizionale, cioè della possibilità che da un nodo del grafo che rappresenta il processo si prenda una direzione piuttosto che un'altra a seguito del verificarsi di una determinata condizione.
- **Azioni.** Se si desidera che una transizione abbia degli *effetti collaterali* di qualsiasi genere, è possibile specificare una azione da invocare quando la transizione viene abilitata o attivata.

La differenza tra le azioni e i predicati di controllo consiste nel fatto che le prime non alterano lo svolgimento del processo, determinando solo effetti collaterali, mentre i secondi hanno lo scopo di definire i percorsi di instradamento tra i gli stati del processo.

In aggiunta ai due precedenti, ci sono ulteriori meccanismi di *callback* nel sistema:

- **Assegnazione.** Servono per determinare a *run-time* l'utente assegnato ad un'attività. Spesso, infatti, chi debba fare cosa viene determinato in base a condizioni che si verificano all'interno dell'applicazione.
- **Temporizzazione.** Le transizioni temporizzate sono quelle che si verificano in base ad una schedulazione predefinita. L'utente può fornire una funzione di callback per determinare l'intervallo di tempo che decide l'attivazione della transizione. In questo modo l'intervallo può essere determinato a *run-time* in base a determinate condizioni. Le *callback* temporizzate vengono eseguite ogni qualvolta una transizione viene abilitata e ritornano una data Oracle valida.
- **Deadline.** Può accadere che un'attività all'interno di un processo debba essere eseguita entro un determinato intervallo di tempo. Può essere invocata a questo scopo una *callback* di *deadline* che deve ritornare la data di *deadline* come data Oracle valida.
- **Timeout.** Quando un utente inizia un'attività, acquisisce un *lock* sulla transizione abilitata e sui *token* che la transizione stessa “consuma”. Si vuole spesso che l'utente non possa tenere occupate tali risorse per un tempo indefinito. E' possibile pertanto definire una *callback* di *timeout*, che rappresenta un intervallo di tempo trascorso il quale l'attività, se non portata ancora a compimento dall'utente, viene cancellata, diventando così disponibile per altri.
- **Notifica.** La *callback* di notifica si verifica ogni qualvolta si debba comunicare ad un utente l'assegnazione di un'attività.
- **Attività non assegnata.** Ogni qualvolta si abilita una transizione, ma non ci sono utenti assegnati, viene tipicamente chiamata questa *callback*, di solito per notificare a qualche responsabile che un'attività non risulta “coperta” da nessun utente.

Di seguito vengono riportati i prototipi delle funzioni di *callback* appena elencate:

#### **Predicati di controllo**

```
function name(
    id_case          in number,
    id_workflow      in varchar2,
    id_transition    in varchar2,
    id_place         in varchar2,
    direction        in varchar2,
    custom_arg       in varchar2
```

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 65 di 80

```
) return varchar2(1);
```

Ritorna 'Y' o 'N' a seconda che l'esito sia positivo o negativo.

### **Azioni**

```
procedure name(  
    id_case          in number,  
    id_transition    in varchar2,  
    custom_arg       in varchar2  
);
```

### **Assegnazione**

```
procedure name(  
    id_task          in number,  
    custom_arg       in varchar2  
);
```

### **Temporizzazione**

```
function name(  
    id_case          in number,  
    id_transition    in varchar2,  
    custom_arg       in varchar2  
)  
    return date;
```

Restituisce la data/ora in corrispondenza della quale la transizione debba essere attivata.

### **Deadline**

```
function name(  
    id_case          in number,  
    id_transition    in varchar2,  
    custom_arg       in varchar2  
)  
    return date;
```

Restituisce la data/ora di *deadline*.

### **Timeout**

```
function name(  
    id_case          in number,  
    id_transition    in varchar2,  
    custom_arg       in varchar2  
)  
    return date;
```

Restituisce la data/ora in corrispondenza della quale il *lock* debba essere rilasciato.

### **Notifica**

```
procedure name(  
    id_task          in number,
```

```
custom_arg      in varchar2,  
role_to         in integer,  
role_from       in out integer,  
subject         in out varchar2,  
body           in out varchar2  
);
```

#### **Attività non assegnata**

```
procedure name(  
    case_id      in number,  
    transition_key in varchar2,  
    custom_arg   in varchar2  
);
```

Per un riferimento dettagliato circa le *API* del modulo di workflow, si rimanda al relativo documento di analisi funzionale "**WAW-WF-AF-05-2001.DOC**".

## **8.2 Modulo "Information Warehouse"**

### **8.2.1 Visione generale**

I servizi applicativi del modulo di *Information Warehouse* vengono utilizzati per la gestione completa dei contenuti all'interno di WEB@WORK.

Il server di contenuti è un componente di base di qualsiasi sito Internet/Intranet. Un elenco dei contenuti necessari all'implementazione dei più diffusi servizi Internet/Intranet richiesti, che non ha pretesa di essere esaustivo, include:

- Database (anagrafiche per il *who's who*, data mart di sintesi di dati statistici, dati a supporto di servizi per il personale, etc.)
- Articoli di stampa (calendari manifestazioni, date di avvenimenti ufficiali, etc.)
- Documentazione (modulistica standard, procedure interne e progetto qualità, organigramma, calendari concorsi, etc.)
- Messaggistica (forum di discussione, news, FAQ, etc.)

Prescindendo dal tipo di origine, sviluppatori, responsabili della pubblicazione e utenti traggono vantaggio dalla gestione dei contenuti attraverso un generico insieme di servizi. Gli sviluppatori ne beneficiano potendo basare le loro applicazioni *content driven* su una singola API, riducendo così la necessità di sviluppi di tipo *custom* (e spesso ridondanti). I *publishers* ne beneficiano perché possono assoggettare tutte le tipologie di contenuto alle medesime pratiche di produzione e gestione, incluso controllo di accesso, classificazione, workflow, etc. Gli utenti ne beneficiano perché possono utilizzare una unica interfaccia per la ricerca, la navigazione e la gestione degli eventuali contributi individuali.

*Information Warehouse* costituisce l'infrastruttura di tutti i servizi applicativi *content driven*. Essa fornisce il seguente insieme di base di servizi:

- Creazione dinamica di nuove tipologie di contenuti con un'interfaccia per la definizione della loro struttura.
- *Storage* comune per tutte le tipologie di contenuti (ciascun contenuto consiste di dati testuali o binari con attributi opzionali derivanti dall'appartenza ad una specifica tipologia).
- Definizione di relazioni tra contenuti di qualsiasi tipo.

- Supporto alla classificazione automatica di contenuti nell'ambito di opportune tassonomie allo scopo di potenziare i meccanismi di ricerca.
- Gestione del *versioning* dei contenuti.
- Interazione consistente con tutti gli altri servizi disponibile nel sistema (permessi, workflow, etc.).
- Motore di ricerca *full-text* per qualsiasi tipologia di contenuto gestito direttamente nel suo formato nativo (documenti Microsoft Office, file PDF, etc.).

E' importante sottolineare come un contenuto sia, utilizzando la terminologia definita descrivendo il *core engine*, un *oggetto* del sistema.

I servizi applicativi di *storage* e, conseguentemente, quelli di *retrieval* devono gestire gli *oggetti contenuto* i qualsiasi formato, testuale e binario. I tipi MIME forniscono un insieme standard di codici per identificare il formato di file di un *oggetto contenuto*. Per motivi di integrità, *Information Warehouse* ha una lista canonica di tipi MIME che possono essere associati al singolo contenuto.

## 8.2.2 Produzione di contenuti per il Web

### 8.2.2.1 Tipologie di contenuti

Una tipologia di contenuto è caratterizzata da un insieme di attributi che possono essere associati ad un *oggetto contenuto* testuale o binario. Ad esempio gli attributi di una notizia da pubblicare sulla Intranet possono includere il titolo, l'autore, l'abstract, il testo e la data di pubblicazione. *Information Warehouse* supporta la memorizzazione di informazioni descrittive per ciascuna tipologia di contenuto:

- Ad una tipologia di contenuto è associata un identificativo alfanumerico univoco (Es. *notizia*) con il quale esso possa essere referenziato.
- Le tipologie di contenuto devono essere associate a delle etichette descrittive (sia al singolare che al plurale) per semplificare il riconoscimento da parte dell'utente.
- Ad una tipologia di contenuto può essere associato un numero qualsiasi di attributi descrittivi.
- Le tipologie di contenuto possono ereditare attributi da altre tipologie (Es. la tipologia *notizia regionale* eredita dalla tipologia *notizia* tutti gli attributi descrittivi, oltre a poterne avere di peculiari).
- Parte della definizione di una tipologia di contenuto può includere una descrizione delle relazioni padre-figlio consentite per contenuti di quel tipo. Ad esempio un contenuto di tipo *notizia* può contenere uno o più contenuti di tipo *immagine*, ma non può contenere contenuti di tipo *curriculum vitae*.
- La definizione di una tipologia di contenuto può includere una lista dei tipi MIME consentiti per contenuti di quel tipo.
- La definizione di una tipologia di contenuto può includere una lista di stringhe alfanumeriche per identificare o marcare relazioni con altri oggetti dello stesso tipo. Ad esempio, la definizione della tipologia *capitolo* per un manuale può includere le stringhe **precedente** e **successivo**.

In aggiunta a quanto specificato, è possibile prevedere che *Information Warehouse* fornisca un supporto nativo alla multilingualità. In questo caso alcuni attributi descrittivi delle tipologie di contenuto potrebbero essere dipendenti dalla lingua (Es. le stringhe **precedente** e **successivo** potrebbero dover avere dei corrispondenti in altre lingue, come **previous** e **next**, etc.).

Più in generale ciò consente di poter pubblicare contenuti localizzati nella lingua prescelta dall'utilizzatore (questo è più utile, probabilmente, in servizi applicativi Internet, rivolti ad una più vasta platea di utilizzatori).

Di seguito viene riportato l'insieme dettagliato dei servizi applicativi di *Information Warehouse* relativamente alle tipologie di contenuto.

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 68 di 80

- Creazione di una nuova tipologia di contenuto, specificando opzionalmente se si tratta dell'estensione di un'altra tipologia (ereditarietà di attributi)
- Creazione, recupero, aggiunta, eliminazione di attributi da una tipologia di contenuto.
- Definizione di un particolare tipo di *workflow* da associare ad istanze di contenuto del tipo in esame.
- Definizione di un insieme di tipi MIME consentiti per la tipologia di contenuto (Es. la tipologia *immagine* accetta solo tipi MIME GIF e JPEG, etc.)
- Definizione di una relazione con un altro tipo di oggetto.

### 8.2.2.2 **Contenuti**

Qualsiasi *contenuto* è un'istanza di una determinata tipologia. Esso è un insieme di dati testuali o binari pubblicabili sul web. Un'istanza di contenuto, all'interno di *Information Warehouse* ha la seguenti caratteristiche:

- Possiede un'identificatore univoco, così da poter essere posta in relazione con altri oggetti del sistema.
- E' costituita da un insieme di attributi e da testo libero o oggetti binari.
- Tra gli attributi ve n'è un insieme (di base) allo scopo di facilitarne la ricerca e lo sviluppo di opportune interfacce di navigazione all'interno di *Information Warehouse*:
  - Titolo.
  - Breve descrizione o sommario.
  - Autore.
  - Data di pubblicazione e/o posting.
  - MIME type.
- Vi sono dei meccanismi per la gestione del *versioning*, (Es. *history* del contenuto).
- Vi sono dei meccanismi per implementare il controllo di accesso per la singola istanza di contenuto tramite associazione di permessi a utenti e/o gruppi.
- Vi sono dei meccanismi per associare il contenuto ad uno o più *workflow* (Es. controllo di convalida per la pubblicazione, etc.).
- Il singolo contenuto può essere esso stesso riguardato come *contenitore* o *padre* per altri contenuti (Es. una notizia può contenere più sezioni).
- Ogni contenuto può essere associato con un numero qualsiasi di oggetti correlati. Il tipo e il numero di relazioni è vincolato a livello di tipologia di contenuto.

Tutti i contenuti saranno organizzati, all'interno di *Information Warehouse* secondo una struttura gerarchica, che ricalca da vicino quella di un normale *file system*. E' quindi previsto un oggetto di tipo *cartella* (analogo di una *directory*).

Di seguito viene riportato l'insieme dettagliato dei servizi applicativi di *Information Warehouse* relativamente ai contenuti:

- Tipi MIME. Possibilità di:
  - Registrare/Deregistrare un tipo MIME.
  - Impostare/Recuperare la descrizione di un tipo MIME.
  - Determinare se un tipo MIME sia testuale o binario.
  - Ottenere una lista dei tipi MIME registrati.
- Localizzazioni. Possibilità di:
  - Aggiungere una nuova lingua di localizzazione
  - Ottenere una lista delle lingue registrate

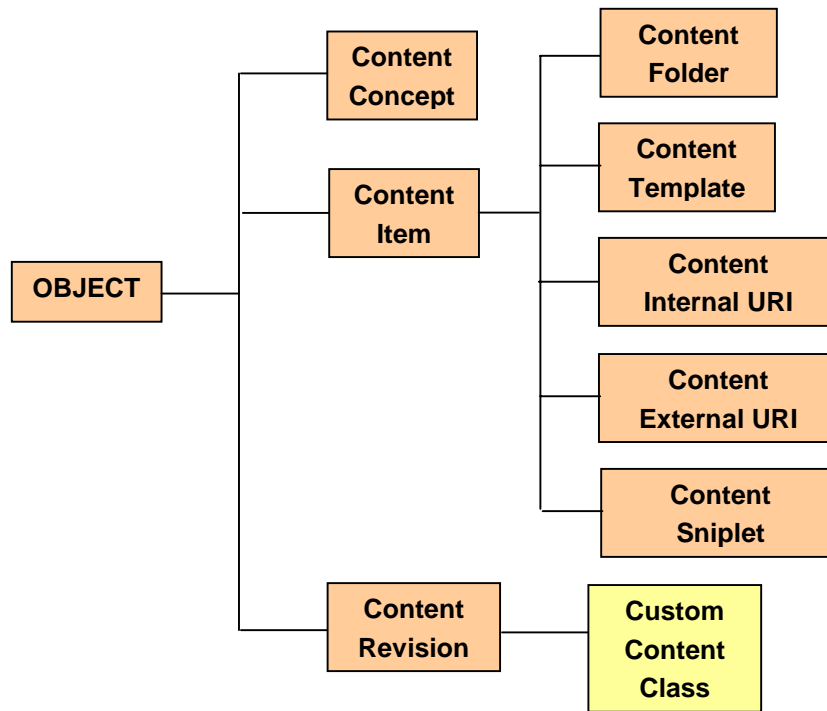


<b>Stato del Documento</b>	<b>Rif.</b>	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 69 di 80

- Oggetti contenuto. Possibilità di:
  - Creare un contenuto, specificando la *cartella* (di default è la *root*).
  - Rinominare un contenuto.
  - Copiare il contenuto in un'altra posizione all'interno del *repository*.
  - Spostare il contenuto in un'altra posizione all'interno del *repository*.
  - Ottenere il percorso completo (in termini di cartelle) del contenuto.
  - Determinare se un contenuto possa contenere a sua volta contenuti di un'altra tipologia specifica, basandosi su quelli eventualmente esistenti e/o sui vincoli imposti a livello di tipologia di contenuto.
  - Stabilire una relazione tra il contenuto ed altri oggetti.
  - Creare una nuova versione del contenuto.
  - Marcare una particolare versione di un contenuto come *live* (in linea).
  - Specificare un intervallo di validità del contenuto.
  - Ottenere una lista delle versioni di un contenuto, inclusi gli utenti coinvolti nelle modifiche, le date di modifica ed eventuali commenti.
  - Creare una nuova versione utilizzando una vecchia come *template*.
  - Classificare un contenuto all'interno di tassonomie e/o mappe concettuali, a livello delle quali poter effettuare le seguenti operazioni di gestione:
    - Creare una classe di contenuti (*concept*), eventualmente come *figlia* di un'altra classe.
    - Utilizzo di un agente software per il supporto alla classificazione automatica.
    - Assegnare/Rimuovere un contenuto ad/da un concetto.
    - Ottenere la lista dei concetti associati al contenuto.
  - Effettuare ricerche, in generale anche *full-text* sui contenuti.
- Cartelle. Possibilità di:
  - Creare una cartella per un raggruppamento logico di contenuti o altre cartelle. Le cartelle possono essere create nella *root* o all'interno di altre cartelle.
  - Impostare/Recuperare l'etichetta e la descrizione di una cartella.
  - Ottenere la lista delle cartelle contenute in una cartella.
  - Spostare una cartella in un'altra cartella.
  - Copiare una cartella in un'altra cartella.
  - Eliminare una cartella solo quando questa sia vuota.

### 8.2.3 Il modello ad oggetti

*Information Warehouse* rappresenta un'estensione del modello ad oggetti del *Core Engine* di [WEB@WORK](#). Il seguente diagramma illustra le relazioni tra le classi di oggetti definite da *Information Warehouse*.



### **Content Concept**

Consente la classificazione di un *item* di contenuto. L'insieme delle istanze della classe *content concept* costituisce la *mappa concettuale* tramite la quale categorizzare i contenuti gestiti. Topologicamente la struttura definita dalle istanze di *content concept* è un grafo.

### **Content Item**

Rappresenta la generica istanza *logica* di un contenuto pubblicato su un sito web. L'istanza *fisica* del contenuto è memorizzata come oggetto *content revision*. Questa implementazione consente il *versioning* degli *item* di contenuto. Ad esempio, se esiste un *item* di contenuto di nome "Curriculum Vitae di Marco Caressa" esisterà anche almeno una revisione associata a questo *item*.

### **Content Revision**

Contiene il dato fisico relativamente ad un *item* di contenuto. C'è una relazione uno-a-molti tra *item* e *revision*. Deve esistere almeno una revisione *live* per ciascun *item*. (La revisione *live* è quella correntemente pubblicata).

### **Custom Content Class**

Rappresenta la generica classe di contenuti definita dall'utente. E' in relazione con le *revisions* per consentire di associare l'istanza della classe *custom* alla revisione, garantendo così il *versioning*.

### **Content Folder**

Contiene *item* correlati e consente ai gestori dei contenuti di raggruppare i contenuti secondo necessità. All'interno di un *folder* gli *items* devono avere un nome univoco.

### **Content Template**

I *template* sono una speciale tipologia di oggetti testuali, utilizzati per specificare la *presentazione* di un contenuto. Essi possono essere associati alle classi di contenuto, significando che ogni contenuto di quel tipo verrà visualizzato utilizzando quel particolare *template*, a meno che lo specifico *item* di contenuto non abbia associato un proprio *template*.

### **Content Internal URI**

Sono puntatori ad *items* all'interno di *Information Warehouse*. Vengono utilizzati per creare dei link tra gli *item* di contenuto.

### **Content External URI**

Sono riferimenti a pagine su altri siti web. Forniscono le basi per la definizione di una gerarchia di *bookmarks* che possano essere gestiti analogamente agli altri *content items*. In particolare, gli *external URI* possono essere marcati con dei *concepts* e correlati a *content items* interni al sito gestito con [WEB@WORK](#).

### **Content Snippet**

Sono una tipologia di oggetti testuali utilizzati per memorizzare componenti di pagina web. *Information Warehouse* utilizza un meccanismo di tag personalizzati da utilizzare all'interno degli *snippet*. Mediante questi si possono generare porzioni di codice HTML partendo da *template* contenenti tali tag personalizzati e memorizzati come *snippet*. L'uso degli *snippet* è indipendente dall'uso dei *template* ed è in ogni caso opzionale.

*Information Warehouse* richiede che lo sviluppatore definisca ciascuna classe di oggetti *contenuto* necessaria. Questa viene definita come qualsiasi altra classe all'interno di [WEB@WORK](#)

Un *item* di contenuto consiste tipicamente di due componenti

- \* Dati testuali o binari memorizzati come singolo oggetto del sistema.
- \* Attributi strutturati memorizzati come valori distinti.

Gli attributi di base per ogni tipologia di contenuto sono memorizzati a livello di oggetti di classe *revision* (in questo modo il valore di tali attributi può essere modificato a livello di singola revisione dell'*item* di contenuto). Tuttavia, la maggior parte dei contenuti richiede anche altri attributi, oltre quelli di base. Ad esempio, quando si memorizza un'immagine si desidera memorizzare anche la larghezza e l'altezza in pixel, così da poterla recuperare meglio o ordinarla rispetto alle altre.

Tali attributi opzionali sono quelli definiti da una classe di oggetti che può essere definita con gli usuali metodi esposti dal *Core Engine*.

## **8.2.4 Organizzazione dei contenuti**

*Information Warehouse* organizza gli *item* di contenuto all'interno di una struttura gerarchica simile ad un file system. I contenuti vengono pertanto gestiti utilizzando delle operazioni di base analoghe.

Subito dopo l'installazione, *Information Warehouse* ha un solo *folder* che rappresenta la *radice* del repository. Gli *items* vengono organizzati creando dei *subfolder* al di sotto della *radice*. E' possibile copiare un *item* da un *folder* ad un altro, così come creare dei link o degli *shortcuts* per

gli *item* in modo da renderli accessibili da altri *folder*. Ogni *item* ha un *nome file* ed un *path* assoluto che lo individua univocamente all'interno del repository.

*Information Warehouse* aggiunge però una ulteriore caratteristica rispetto ad un file system tradizionale. Ogni *content item* (non solo un *folder* quindi) può costituire un contenitore per qualsiasi numero di altri *content items*. Si immagini ad esempio un libro costituito da una prefazione, un certo numero di capitoli ed una bibliografia (che a loro volta possono avere una ulteriore strutturazione). Il libro in sé è un *content item*, poiché è caratterizzato da attributi (editore, data di pubblicazione, etc.) ma è al contempo un contenitore logico di altri *items*.

E' importante notare come un *folder* sia solo uno speciale tipo di *content item*. La relazione gerarchica tra un *folder* e gli *items* in esso contenuti non è dissimile da quella che lega un libro ai suoi capitoli. I *folder* possono essere riguardati come generici contenitori per raggruppare degli *items* che non necessariamente siano componenti strutturali di un altro *item*.

Di default, *Information Warehouse* ha una radice nel file system virtuale per i *content items* ed una radice per i *templates*. Talvolta non è sufficiente. Si supponga che un *package* possa venire istanziato diverse volte e che questo debba memorizzare i contenuti relativamente ad ogni istanza. In questo caso si potrebbe dover creare una radice relativa al *package* in modo che i contenuti di un'istanza non interferiscano con quelli di un'altra.

### 8.2.5 Relazioni tra contenuti

La maggior parte delle applicazioni *content-based* richiede che i contenuti possano essere messi in relazione tra loro o con altre classi di oggetti. Esempi di ciò possono essere i seguenti:

- \* Una notizia può essere *linkata* ad altre notizie nell'ambito della stessa categoria.
- \* Un articolo può essere correlato ad una serie di foto e/o grafici da includere nell'articolo stesso.
- \* Un testo particolarmente lungo può essere suddiviso in sezioni, ognuna delle quali debba essere visualizzata separatamente.
- \* Una recensione su un prodotto può essere collegata al prodotto stesso.
- \* La foto di un utente può essere collegata alla scheda informativa dell'utente medesimo.

I servizi che consentono di definire e gestire relazioni tra oggetti e classi di oggetti sono messi a disposizione dal *Core Engine*. Gli sviluppatori che debbano risolvere situazioni molto specifiche e debbano o vogliano sviluppare una loro propria interfaccia verso *Information Warehouse* (diversa da quella fornita con il sistema) possono utilizzare direttamente le API del *Core Engine*.

*Information Warehouse* fornisce una sua propria infrastruttura per la gestione di relazioni tra *content item* che utilizza in modo trasparente i servizi del *Core Engine* ma che non obbliga lo sviluppatore a farvi ricorso direttamente, almeno per le situazioni di uso più comune.

I meccanismi di relazione tra contenuti descritti di seguito potrebbero non essere sufficienti per alcune necessità specifiche. Ad ogni modo, poiché si tratta di *oggetti* del sistema, gli sviluppatori hanno sempre la possibilità di estendere le classi relative così come per qualsiasi altro oggetto.

#### 8.2.5.1 Relazioni padre-figlio

Spesso un contenuto può essere utilizzato come naturale contenitore per altri contenuti. Tali relazioni *padre-figlio* sono gestite dalla organizzazione gerarchica di base implementata dal file system virtuale di *Information Warehouse*.

Talvolta è necessario vincolare il numero e la tipologia dei contenuti dei contenuti *figli*. Ad esempio una notizia può avere al massimo una foto, un report strutturato può avere al massimo tre sezioni, e così via. Inoltre, può essere necessario classificare o identificare contenuti *figli* dello stesso tipo. Ovviamente, rifacendosi all'esempio precedente, le tre sezioni del report saranno caratterizzate da un ordine logico con il quale esse dovranno essere presentate all'utente.

*Information Warehouse* fornisce un modello di dati e delle opportune API per la definizione e la gestione di tali relazioni e dei vincoli relativi.

### 8.2.5.2 Relazioni contenuto-oggetto

Oltre alla relazione gerarchica implementata in modo nativo dal sistema di *folder* virtuali, i contenuti possono essere associati con dei *link* a qualsiasi altro oggetto all'interno del sistema (prodotti per un'applicazione di *e-commerce*, utenti del sistema o contenuti di tematiche correlate). Gli stessi tipi di vincoli descritti nella relazione di tipo *padre-figlio* devono poter essere definiti anche per le relazioni tra contenuti ed oggetti in generale. *Information Warehouse* fornisce un modello di dati e delle opportune API per la definizione e la gestione di tali relazioni e dei vincoli relativi.

### 8.2.6 Presentazione dei contenuti

*Information Warehouse* consente di associare dei *template* sia alle classi di contenuto che agli specifici contenuti. Un *template* specifica come un contenuto debba essere visualizzato quando venga esportato su file system o *servito* direttamente al *client*.

*Information Warehouse* non fa alcuna assunzione circa il sistema di *templating* utilizzato dall'application server che viene utilizzato per far girare il sistema. I *template* sono semplicemente messi a disposizione dell'application server come oggetti testo ed esso ha la responsabilità di effettuare il *merging* tra il *template* e il contenuto per generare dinamicamente la pagina web.

Gli *item* di contenuto e i *template* sono organizzati in due gerarchie separate all'interno di *Information Warehouse*. Ciò consente molto facilmente di predisporre più *template* altrettante diverse visualizzazioni del medesimo contenuto. Inoltre assicura che il codice del *template* non sia mai accessibile tramite una URL pubblica.

### 8.2.7 Pubblicazione dei contenuti

#### 8.2.7.1 Generalità

*Information Warehouse* non pone alcuna restrizione circa il metodo impiegato per la distribuzione delle pagine. Le applicazioni sono libere di effettuare le opportune query sul *repository* tramite le API e di processare i dati nel modo desiderato.

Benché non vi siano restrizioni circa il metodo di pubblicazione, le API di *Information Warehouse* intendono facilitare un generico meccanismo di pubblicazione *template-based*, indipendentemente dallo specifico strato di presentazione utilizzato.

#### 8.2.7.2 Workflow di pubblicazione

*Information Warehouse* implementa un *workflow* di pubblicazione basato sull'utilizzo dei servizi applicativi esposti dal modulo *Workflow*.

La maggior parte degli schemi di pubblicazione segue, con qualche variazione, un *workflow* di questo tipo:

Stato	Task	Descrizione
Creato	<b>Authoring</b>	L'autore ha creato il <i>content item</i>
Terminato	<b>Editing</b>	L'autore ha prodotto il <i>content item</i>
Approvato (pronto per la pubblicazione)	<b>Publishing</b>	L'editore ha approvato il <i>content item</i>

Pubblicato	Nessuno	Il <i>publisher</i> ha pubblicato il <i>content item</i>
------------	---------	--

In qualsiasi momento del *workflow*, un utente cui sia stato assegnato un *task* relativamente ad un *content item* deve essere in grado di effettuare un'operazione di *check-out*, che consente al sistema di notificare agli altri utenti che la risorsa è bloccata ed in lavorazione. Per rendere nuovamente disponibile la risorsa è necessario effettuare un'operazione di *check-in*. In questo caso l'utente dovrebbe avere a disposizione tre opzioni:

- \* Effettuare il *check-in* dell'*item* ma non marcare il *task* come *terminato* (consentendo così a qualcun altro, che abbia magari il medesimo ruolo, di lavorare al *task*). Il *task* correntemente abilitato (authoring, editing o publishing) non viene modificato.
- \* Effettuare il *check-in* dell'*item* e muoverlo al *task* successivo. Per il *task* di *authoring* ciò significa che il *task* è completo. Per i *task* successivi significa l'approvazione dell'*item*.
- \* Effettuare il *check-in* dell'*item* e muoverlo al *task* precedente, con ciò significandi il rifiuto dell'*item*.

### 8.2.8 Modello dei dati e interfacce

Per un riferimento dettagliato circa il modello dei dati e le *API* del modulo di *Information Warehouse*, si rimanda al relativo documento di analisi funzionale "**WAW-IW-AF-05-2001.DOC**".

## 8.3 Modulo "Messaging"

I servizi applicativi del modulo di *messaging* forniscono la possibilità di memorizzare, inviare e ricevere messaggi nell'ambito dell'applicazione web. Lo scopo è quello di dare supporto a diverse tipologie di messaggi attraverso un'unica infrastruttura centralizzata.

Il modulo *messaging* utilizza uno *storage* centralizzato per le varie tipologie di messaggi. Un messaggio è caratterizzato da un tipo MIME specificato, oltre che al contenuto vero e proprio. Ai messaggi possono essere associati degli *allegati* (*attachments*), ognuno dei quali caratterizzato da un tipo MIME.

Il messaggio è relato all'*attore* che lo invia e a quello che lo riceve. Inoltre è disponibile un meccanismo di *threading*, cioè di catene di messaggi costituite da un *albero* di risposte ad un messaggio di partenza.

Sui messaggi è possibile effettuare delle ricerche, anche *full-text*.

Il modulo deve fornire i seguenti servizi applicativi per la manipolazione dei messaggi:

- o Creazione/modifica/visualizzazione.
- o *Versioning* (ci deve essere la possibilità di creare diverse versioni di un messaggio).
- o Eliminazione (possibilità di eliminare tutte le versioni e gli allegati di un messaggio in cascata).
- o Associazione di allegati (con definizione, per essi, dei tipi MIME).

Circa l'invio dei messaggi dovrà essere possibile utilizzare, oltre al canale tradizionale e-mail, anche altri canali (SMS e , in futuro, UMTS, etc.). A tale scopo è sufficiente che il sistema esponga un'interfaccia standard per l'esportazione dei messaggi.

Per la realizzazione di tale interfaccia, la tecnologia di riferimento che proponiamo di utilizzare, e che consente di isolare il contenuto dalle modalità di rappresentazione è XML.

Per canali di output diversi dall'e-mail è sufficiente realizzare un accordo con un *carrier* che provvederà all'instradamento del messaggio secondo il canale prescelto. La maggior parte dei *carrier* richiedono una rappresentazione XML dei messaggi da inviare.

<u>Stato del Documento</u>	<u>Rif.</u>	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 75 di 80

L'invio dei messaggi deve poter riguardare un singolo messaggio, così come un insieme di messaggi da raggruppare in un *digest* da inviare ad un singolo utente o a gruppi di utenti.

### 8.3.1 *Strutture di dati e interfacce*

Per un riferimento dettagliato circa il modello dei dati e le *API* del modulo di *Messaging*, si rimanda al relativo documento di analisi funzionale "**WAW-ME-AF-05-2001.DOC**".



## 9 La realizzazione del sistema

Di seguito viene riportato un quadro di sintesi delle linee di attività che costituiscono oggetto dello sviluppo di WEB@WORK.

Tali attività sono relative alla realizzazione dell'infrastruttura di *back-end* del sistema, costituita dai database di supporto al *Core Engine* e ai moduli ipotizzati e dalle API, sviluppate in PL/SQL, che ne costituiscono le interfacce applicative. Non sono al momento oggetto di stima attività relative alla realizzazione di componenti di interfaccia grafica (*front-end*).

Va tuttavia fatta una distinzione tra interfaccia grafica di applicazioni web sviluppate utilizzando [WEB@WORK](#) e l'interfaccia di amministrazione del sistema. Le prime sono una componente delle cosiddette "verticalizzazioni" e quindi andranno realizzate di volta in volta, la seconda è certamente "inclusa" nel *framework*, anche se non ne viene di seguito stimato lo sforzo realizzativo.

Alcuni esempi di verticalizzazioni potrebbero essere costituiti in prima istanza da un'applicazione di gestione documentale (l'analogo dell'*Archdoc* attuale, con in più i servizi attualmente non disponibili che saranno invece forniti dalla piattaforma di WEB@WORK - agente di classificazione automatica, migliore *storage* dei documenti, *workflow* di gestione degli stessi, etc.) e da un'applicazione di *content management system* a supporto di un generico portale web. In tal caso si svilupperanno delle componenti di interfaccia che, soprattutto per le funzionalità di amministrazione, potranno essere standardizzate.

Per ciò che riguarda lo sviluppo di quella che in gergo viene definita applicazione di *content delivery*, vale a dire l'interfaccia web percepita dagli utenti del generico sito/portale, si anticipa sin d'ora la necessità di sviluppare un ulteriore modulo di WEB@WORK relativo al *templating* di pagine web. Con questo termine si intende la possibilità di definire dei modelli di pagina HTML contenenti, mediante espressione in una opportuna sintassi, dei costrutti atti a richiamare i contenuti tramite i servizi forniti dal modulo di *Information Warehouse* per la generazione dinamica di pagine web.

Un'ulteriore attività, che però riguarda in prima battuta l'infrastruttura attuale di *Archdoc* e successivamente anche WEB@WORK, è costituita dal *porting* su piattaforma *JSP* dell'attuale applicazione web e delle varie personalizzazioni in corso di sviluppo (tali personalizzazioni interessano sia progetti di Engiweb.com in fase di definizione, che i progetti di ricerca europei ECOLNET ed EUSLAND, citati nella prima parte di questo documento e gestiti dal Laboratorio di Ricerca e Sviluppo di Engineering).

Tale *porting* verso *JSP* ha due obiettivi:

- Svincolare l'infrastruttura attuale e futura dall'utilizzo di Oracle Application Server, che attualmente è una componente architetture imprescindibile.
- Realizzare una soluzione con una tecnologia che rappresenta uno standard *de facto*.

Di seguito l'elenco delle attività previste:

- **Pianificazione ed analisi:** In questa linea rientrano le attività di analisi dettagliata delle funzionalità previste da WEB@WORK, allo scopo di strutturare il *framework* coerentemente con gli obiettivi strategici
  - **Core Engine**
    - Definizione
    - Analisi dati (strutture di rappresentazione)
    - Analisi funzionale (interfacce)



<b>Stato del Documento</b>	<b>Rif.</b>	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 77 di 80

- Pianificazione integrazione con sistema di *knowledge representation* approntata dal Laboratorio di Ricerca e Sviluppo
- **Modulo Information Warehouse**
  - Definizione
  - Analisi dati (strutture di rappresentazione)
  - Analisi funzionale (interfacce)
- **Modulo Workflow**
  - Definizione
  - Analisi dati (strutture di rappresentazione)
  - Analisi funzionale (interfacce)
  - Pianificazione integrazione agente di classificazione intelligente
- **Modulo Messaging**
  - Definizione
  - Analisi dati (strutture di rappresentazione)
  - Analisi funzionale (interfacce)
- **Realizzazione**
  - **Core Engine**
    - Disegno database
    - Sviluppo API PL/SQL
      - *Integrazione con API di knowledge representation approntata dal Laboratorio di Ricerca e Sviluppo*
    - Produzione documentazione
    - Test
  - **Modulo Information Warehouse**
    - Disegno database
    - Sviluppo API PL/SQL
    - Integrazione agente di classificazione intelligente
    - Produzione documentazione
    - Test
  - **Modulo Workflow**
    - Disegno database
    - Sviluppo API PL/SQL
    - Produzione documentazione
    - Test
  - **Modulo Messaging**
    - Disegno database

- Sviluppo API PL/SQL
- Produzione documentazione
- Test
- **Coordinamento e controllo**
  - **Coordinamento generale progetto**
  - **Controllo avanzamento lavori**

## 9.1 Stime di massima

Di seguito l'elenco delle attività accorpate relative all'analisi, realizzazione e coordinamento di WEB@WORK, corredate della stima prevista in termini di giorni/uomo. Per una corretta valutazione dei tempi "elapsed" di ciascuna fase si rimanda al diagramma a pagina seguente. Circa le stime in termini di costi, si rimanda al relativo modulo C12 "ECW-DB-API-C12.XLS".

Ad ogni modo l'inizio delle attività è previsto entro la terza decade di aprile 2001. Per ciò che riguarda il completamento si ipotizza la terza decade di giugno 2001 per il *Core Engine* e il modulo *Information Warehouse*, la terza decade di luglio per il modulo di *Workflow* e la terza decade di settembre per il modulo di *Messaging*:

• Coordinamento e controllo avanzamento lavori	→	<b>60 giorni/uomo</b>
• Pianificazione ed analisi ( <b>Tot. 90 giorni/uomo</b> )		
○ Core Engine	→	30 giorni/uomo
○ Modulo Information Warehouse	→	25 giorni/uomo
○ Modulo Workflow	→	25 giorni/uomo
○ Modulo Messaging	→	10 giorni/uomo
• Realizzazione ( <b>Tot. 180 giorni/uomo</b> )		
○ Core Engine	→	60 giorni/uomo
○ Modulo Information Warehouse	→	50 giorni/uomo
○ Modulo Workflow	→	50 giorni/uomo
○ Modulo Messaging	→	20 giorni/uomo
• Produzione documentazione ( <b>Tot. 40 giorni/uomo</b> )		
○ Core Engine	→	15 giorni/uomo
○ Modulo Information Warehouse	→	10 giorni/uomo
○ Modulo Workflow	→	10 giorni/uomo
○ Modulo Messaging	→	5 giorni/uomo
• Test (complessivo)	→	<b>30 giorni/uomo</b>
<b>Totale</b>	→	<b>400 giorni/uomo</b>

## 10 Appendice

Di seguito l'elenco dei termini anglofoni e degli acronimi utilizzati nel documento:

- **Active-x** componenti software che possono essere eseguiti all'interno di un contenitore (Es. Microsoft Internet Explorer).
- **API (Application Program Interface)** insieme di funzioni che consentono di rendere programmabile il comportamento di una componente software.
- **Application server** componente software che esegue in maniera centralizzata un'applicazione su richiesta di utenti connessi lato *client*.
- **Backup-recovery** funzionalità per il salvataggio di sicurezza dei dati e per il loro ripristino in caso di guasti al sistema informatico.
- **Web based** dicesi di tecnologie e/o prodotti informatici che utilizzino le infrastrutture tecnologiche e i protocolli di comunicazione di Internet.
- **Business** attività, possibilità di profitto.
- **Business-logic** insieme della logica applicativa di una soluzione software.
- **Bug Tracking** tracciamento e gestione degli errori in un prodotto software.
- **Callback** meccanismo di "ritorno di chiamata" da una procedura alla procedura chiamante. In **WEB@WORK** viene utilizzato per consentire l'esecuzione di codice personalizzato.
- **Client side** dicesi di componenti o tecnologie relative alla macchina su cui opera l'utente finale di un sistema informativo.
- **Content Management** complesso delle attività volte a gestire i contenuti informativi, dalla produzione attraverso la validazione sino alla pubblicazione degli stessi.
- **Deadline** data entro la quale una determinata attività deve giungere a completamento.
- **Download** scarico di file da un server tramite il web.
- **E-business** complesso delle attività relative alla sopravvivenza di un'organizzazione effettuato su un'infrastruttura di rete
- **FAQ** (Frequently Asked Questions) liste di domande frequenti su una determinata tematica.
- **Folder** cartella, directory
- **Knowledge** termine per indicare il patrimonio di conoscenza di una persona o organizzazione
- **Knowledge Management** sistemi informatici per la gestione della conoscenza sotto forma di informazioni strutturate e non.
- **Knowledge Workers** termine che indica lavoratori che fanno del patrimonio di conoscenza la materia prima della loro attività
- **ICT** acronimo per Internet and Communication Technology
- **Information technology** settore delle tecnologie informatiche.
- **Item** elemento (Es. *content item* => elemento di contenuto, etc.)
- **Java Virtual Machine** software che consente l'esecuzione di codice bytecode java.
- **Lock** meccanismo di blocco di dati relativo all'utilizzo, da parte di un utente, dei dati medesimi.
- **Messaging and queuing** sistemi di gestione di messaggi anche tra sistemi eterogenei.
- **Newsgroups** gruppi di discussione per la condivisione di informazioni e messaggistica su determinate tematiche.

Stato del Documento	Rif.	
Versione 1.0	<a href="mailto:marco.caressa@engiweb.com">marco.caressa@engiweb.com</a>	Pagina 80 di 80

- **Open source** possibilità di riutilizzare e distribuire liberamente il codice sorgente di componenti software all'interno di proprie applicazioni, garantendo a terzi l'utilizzazione e la distribuzione dei propri sorgenti.
- **Override** meccanismo di ridefinizione in un ambito più ristretto.
- **Ownership** proprietà, appartenenza
- **Porting** implementare una soluzione software esistente su una differente piattaforma tecnologica.
- **RDBMS** (Relational Database Management System) Sistema per la gestione di *database* relazionali.
- **Repository** contenitore logico di informazioni.
- **Ricerche full-text** possibilità di effettuare ricerche su informazioni non strutturate (tipicamente testuali).
- **Run-Time** dicesi di azioni, eventi o situazioni che hanno luogo durante l'esecuzione di un programma.
- **Semantic-based:** letteralmente "basato sulla semantica". Il termine é utilizzato per caratterizzare delle applicazioni o funzioni implementate sfruttando l'analisi semantica di testo in linguaggio naturale o strutture ontologiche
- **Server Pages** sistemi per la generazione dinamica di pagine web tramite pre-elaborazione sul server (Es. Microsoft ASP, PHP, JSP, etc.)
- **Servlet** programmi eseguiti lato server da un opportuno motore (*Servlet Engine*) che possono essere invocati via protocollo HTTP.
- **Stored procedures** programmi memorizzati all'interno di un DBMS.
- **Upload** invio di file tramite il web sul server.
- **Web publishing** possibilità di pubblicare su un sito web informazioni in maniera dinamica e controllata.
- **Workflow** ciclo di vita di un'informazione.
- **Wrapper** dicesi di una componente software che ha il compito di realizzare esclusivamente un'interfaccia per un'altra componente.