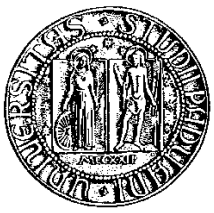

**Università degli Studi di Padova**

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingeneria Informatica

Tesi di laurea

# ANALISI E REALIZZAZIONE DI PROCEDURE PER IL WORKFLOW MANAGEMENT IN UN'AZIENDA CERTIFICATA ISO:9001

**Relatore:** Ch.mo Prof. Matteo Bertocco
**Correlatore:** Ing. Mauro Franchin

**Laureando:** Enrico Righetto

19 Aprile 2010

*Ai miei genitori*

# Prefazione

Questa tesi è il risultato di un lavoro svolto da due studenti del corso di Laurea Specialistica in Ingegneria Informatica che pone come obiettivo la realizzazione di un sistema di *WorkFlow Management* utilizzabile da aziende che lavorano in conformità alle norme ISO 9001 o che hanno come obbiettivo tale certificazione. In particolare, il sistema deve essere capace di adattarsi il più possibile a tutte le realtà aziendali, assimilare la struttura dell'azienda e, su di essa, applicare l'insieme delle regole che permettano di raggiungere la certificazione.

Portare un'azienda ad ottenere la certificazione ISO 9001 è un'operazione che richiede un avanzamento per passi successivi. Il lavoro descritto è il frutto di una collaborazione tra un'azienda padovana che intende intraprendere il percorso di certificazione e l'Università degli Studi di Padova.

Il lavoro è stato svolto in team sin dalle prime fasi, con un carico di lavoro equamente distribuito, sotto la supervisione del relatore. Quanto riportato nel presente elaborato riflette il modo di operare. Non essendo possibile produrre una suddivisione netta nell'esposizione dei temi unicamente riferibile ad un unico autore si è scelto di riportare in entrambe le tesi i contenuti oggetti di lavoro comune. Per queste ragioni, i contenuti di questo lavoro sono pressochè duplicati nella tesi:

> "*Progettazione e implementazione di procedure di workflow management in un'azienda certificata ISO:9001*", di Molinaroli Andrea.

La prefazione della suddetta tesi riporta una nota analoga alla presente.

*Enrico Righetto*
*Andrea Molinaroli*

# Indice

# Capitolo 1

# Introduzione

Sempre più spesso le aziende, per poter far fronte alla concorrenza, devono offrire ai clienti certezze sulla qualità dei prodotti sviluppati e trasparenza nei processi aziendali.

La necessità di avere un riferimento internazionale, attraverso il quale stabilire la *qualità dei prodotti*, ha determinato, nel 1987, l'emanazione, da parte dell'ISO[1] (International Organization for Standardization), di una famiglia di *norme internazionali*. Tale famiglia è meglio conosciuta con il nome di ISO 9001 e contiene un insieme di regole il cui scopo è garantire che un'azienda implementi un sistema di gestione interna in grado di garantire la qualità dei prodotti.

Sebbene la volontà di intraprendere un percorso di certificazione, con lo scopo di beneficiarne, sia spesso dettata da logiche di mercato, va ricordato che rivedere l'*asset* aziendale per renderlo adatto allo standard ISO, e il lavoro per mantenerlo tale, può, spesso, richiedere un'overhead notevole ai dipendenti aziendali. Per fronteggiare tale problema vengono, generalmente, impiegati strumenti al fine di automatizzare il più possibile il lavoro di produzione della documentazione necessaria. Le aziende si trovano, quindi, costrette ad adottare strumenti, in particolare software, che spesso impongono la loro struttura, la quale obbliga chi li utilizza a modificare, anche pesantemente, il proprio modo di lavorare.

Questa tesi è il risultato di un lavoro svolto da più studenti del corso di Laurea Specialistica in Ingegneria Informatica che pone come obiettivo la realizzazione di un sistema di *WorkFlow Management* utilizzabile da aziende che lavorano in conformità alle norme ISO 9001 o che hanno come obbiettivo tale certificazione. In particolare, il sistema deve essere capace di adattarsi il più possibile a tutte le realtà aziendali, assimilare la struttura dell'azienda e, su di essa, applicare l'insieme delle regole che permettano di raggiungere la certificazione.

Come si vedrà, portare un'azienda ad ottenere la certificazione ISO 9001 è un'operazione che richiede un avanzamento per passi successivi. Il lavoro descritto nella presente tesi è il frutto di una collaborazione tra un'azienda padovana

[1] `http://www.iso.org`

che intende intraprendere il percorso di certificazione e l'Università degli Studi di Padova.

La prima parte della tesi presenta un'introduzione alle normative e descrive la struttura aziendale mettendola in relazione con i requisiti prescritti dalle norme stesse. In particolare, nel capitolo 2 viene presentata la normativa ISO 9001. Il capitolo 3, invece, descrive le norme ISO/IEC TR 15504 Information Technology – Software Process Assessment. Esse sono norme specifiche che forniscono un modello per la valutazione di processi software. Il capitolo 4 presenta il profilo dell'azienda e il modello per la descrizione e la rappresentazione dei processi, a loro volta descritti. Il capitolo 5, infine, raccoglie il lavoro svolto nell'ambito del corso di Ingegneria della Qualità. La norma ISO/IEC TR 15504 fornisce un modello standard per i processi delle aziende IT, utilizzando il quale è possibile effettuare un'operazione di *Capability Determination*. Il lavoro si è basato nel confronto tra i processi aziendali e quelli forniti dal modello, in modo da verificare in che modo essi adempiono alle prescrizioni imposte e per evidenziare le carenze che devono essere necessariamente colmate per poter aspirare ad ottenere la certificazione. A seguito di questa prima analisi, l'azienda può, quindi, provvedere ad adattare (se presenti) e ad implementare (se assenti) quei processi che evidenziano particolari violazioni della norma o che, per vari motivi, si sia ritenuto di dover modificare.

La seconda parte del lavoro è stata, invece, dedicata alla progettazione e al conseguente sviluppo di un software per il sistema di gestione della qualità (SGQ) che rispetti i requisiti normativi precedentemente individuati. Il capitolo 6 presenta il progetto WQF (acronimo di *Work-Quality-Flow*), evidenziandone le idee di fondo e gli obbiettivi prefissati. Nel capitolo 7 sono descritte le fasi per la definizione del piano di lavoro. L'analisi funzionale del software e l'analisi tecnica, in cui viene presentata l'architettura dell'applicazione e le tecnologie utilizzate, sono presentate rispettivamente nei capitoli 8 e 9. Nel capitolo 10 viene descritta, sotto forma di tutorial, la realizzazione di una semplice applicazione di prova per sperimentare i meccanismi di autenticazione e gestione dei permessi. La struttura della base di dati su cui lavora l'applicazione è riportata nel capitolo 11. I capitoli 12, 13 e 14, infine, contengono la manualistica per l'installatore, il disegnatore dei processi e l'utente abituale.

L'appendice A riporta i sorgenti e la struttura delle directory dell'applicazione di prova sulla sicurezza. I sorgenti dell'applicazione WQF sono stati volutamente ommessi per ragioni di spazio: essi sono comunque disponibili nel disco allegato.

# Parte I

# Introduzione alle normative e caso aziendale

# Capitolo 2

# La Norma ISO 9001

## 2.1 La norma e i suoi obbiettivi

Nell'ultimo ventennio il mondo delle imprese, sia manifatturiere che di servizio, sia pubbliche che private, è stato protagonista di una vera e propria rivoluzione della qualità che ha profondamente influenzato le strategie d'impresa, il management, il ruolo delle persone e il quotidiano approccio alle attività aziendali. Questa rivoluzione globale ha, per le aziende, un passaggio cruciale, che in molti casi diviene obbligato per la sopravvivenza delle stesse nel mercato: instaurare un Sistema di Gestione per la Qualità (SGQ) in grado di essere certificato da un ente accreditato di parte terza.

L'ISO[1] (International Organization for Standardization) e il CEN[2] (European Committee for Standardization) hanno redatto le regole organizzative minimali cui le aziende devono conformarsi per garantire in modo continuo e costante la qualità di prodotti e/o servizi; queste regole sono rappresentate dalle norme UNI EN ISO 9000. Tali norme non fanno riferimento a particolari settori industriali e commerciali, ma sono applicabili a qualunque attività e a tutti i tipi di impresa e di organizzazione, qualsiasi sia la loro dimensione e il livello di tecnologia impiegata. Le norme UNI EN ISO 9000, recepimento italiano delle norme

[1] ISO è un'associazione mondiale di organismi nazionali di formazione, i cui comitati tecnici effettuano l'elaborazione delle norme internazionali. Ogni organismo nazionale di formazione interessato a un argomento per il quale è stato insediato un comitato tecnico ha il diritto di essere rappresentato in tale comitato. Partecipano ai lavori anche le organizzazioni internazionali di estrazione governativa e non, che intrattengono rapporti con l'ISO. L'ISO collabora strettamente con l'IEC (International Electrotechnical Commission) in tutti i campi di formazione del settore elettrotecnico.

[2] CEN (Comitato Europeo di Normazione) è un ente normativo che ha lo scopo di armonizzare e produrre norme tecniche in Europa in collaborazione con enti normativi nazionali e sovranazionali, quali per esempio l'ISO. Il CEN lavora in accordo con le politiche dell'Unione Europea e dell'EFTA (European Free Trade Association) per favorire il libero scambio e la tutela dei consumatori, dei lavoratori e dell'ambiente, producendo standard che sono poi normalmente recepiti dagli enti nazionali dei singoli paesi.

ISO 9000, forniscono alle aziende italiane un pacchetto di regole riguardanti la conduzione aziendale per la qualità e l'assicurazione della stessa.

L'UNI (Ente Nazionale Italiano di Unificazione) definisce la norma un *"documento prodotto mediante consenso di tutte le parti interessate e approvato da un organismo riconosciuto, che fornisce, per usi comuni e ripetuti, regole, linee guida o caratteristiche relative a determinate attività o ai loro risultati, al fine di ottenere il miglior ordine in un determinato contesto"*. Gli obiettivi fondamentali della normazione sono:

- la realizzazione di un mezzo chiaro e univoco di espressione di comunicazione fra tutte le parti interessate all'interno dell'organizzazione;
- il miglioramento dell'economia generale, attraverso la razionalizzazione della produzione di materiali grezzi, semilavorati e finiti;
- la salvaguardia della salute e della sicurezza degli individui e la tutela dell'ambiente;
- la protezione del consumatore mediante un livello di qualità dei prodotti e dei servizi debitamente controllato e adeguato alle sue necessità.

Alla base della definizione di Sistema di Gestione per la Qualità c'è il concetto di sistema: esso è un insieme di oggetti (parti, componenti, funzioni, ecc.) legati tra loro da relazioni di interdipendenza. Per interdipendenza s'intende che un intervento effettuato esclusivamente su una singola parte del sistema ha ripercussioni anche sulle altre; perciò quando si parla di sistema è necessario un approccio globale e mai parziale. Con l'approccio sistemico non si vuole certo sminuire l'importanza dei singoli componenti, ma si cerca piuttosto di considerarli e di studiarli in ragione del loro essere parti del tutto. Per ottenere questo importante risultato è necessario pianificare e organizzare le attività, i ruoli, le responsabilità e i supporti operativi in modo da coinvolgere tutti i funzionari aziendali. Un'azienda che operi in ottica di qualità e che abbia una gestione per processi vede all'interno di essi una sequenza di attività, ciascuna delle quali fornisce a quella successiva il proprio prodotto,ovvero l'attività a monte viene vista come un vero e proprio fornitore mentre l'attività a valle come un cliente, lungo una catena di processi interni che inizia con l'identificazione delle esigenze e delle aspettative del cliente (esterno) e si conclude con il suo soddisfacimento.

In seguito a questa interpretazione, la qualità ha assunto le prerogative di una filosofia che coinvolge tutti gli attori dell'organizzazione e tutti i processi che ne ottimizzino le risorse, diventando totale. Una qualità così intesa, e considerata dall'azienda come costante ricerca delle caratteristiche più importanti da individuare e fare proprie, si compone di tre caratteristiche principali: quella operativa (il prodotto/servizio è efficiente e affidabile); quella relativa alle caratteristiche tecniche (le prestazioni del prodotto/servizio sono frutto di indagini di marketing, analisi di mercato e valutazioni sulla soddisfazione del cliente);

quella dei servizi connessi al prodotto/servizio (sostituzione, tempi e modalità di consegna, forme di pagamento e assistenza post-vendita). E' proprio su aspetti quali la qualità dei servizi post-vendita che, in un mercato attestato su elevati standard produttivi, si va sempre più concentrando l'attenzione: trasporto, montaggio e manutenzione sono elementi fondamentali e possono determinare il consolidamento o la perdita di clientela.

Ai fini di una loro maggiore spendibilità operativa, si riassumono i concetti base per il raggiungimento del Total Quality Management (TQM) in dieci punti sequenziali:

1. la priorità assoluta dell'azienda, cioè la condizione essenziale per garantirne la sopravvivenza, è costituita dal cliente, senza il quale l'azienda non ha ragione di esistere;

2. in quest'ottica il tipo di cliente più importante è il cliente consolidato. Un fatturato realizzato con clienti consolidati è molto più sicuro di quello realizzato con clienti occasionali;

3. un cliente diventa consolidato se è soddisfatto del precedente acquisto. La soddisfazione del cliente diventa quindi la vera priorità operativa dell'azienda;

4. il profitto è il premio di questa soddisfazione, il fatturato ne è la sua misura;

5. la soddisfazione del cliente si ottiene essenzialmente fornendogli un prodotto o un servizio di alta qualità: è proprio la qualità di quanto ha già acquistato il fattore che più ne condizionerà il prossimo acquisto (il prezzo è, invece, un fattore prioritario per l'acquisizione di un nuovo cliente). La qualità assume un significato più ampio e contiene tutto ciò che garantisce la soddisfazione del cliente;

6. per garantirsi il consolidamento del cliente occorre assicurarsi la sua continua soddisfazione a ogni successivo acquisto. Questo risultato non è ottenibile semplicemente fornendo un elevato grado di qualità e mantenendolo costante nel tempo. La soddisfazione presume il miglioramento. Solo miglioramenti continui del prodotto e/o del servizio fornito possono garantire un elevato grado di soddisfazione del cliente, in modo tale da condizionarlo positivamente per il successivo acquisto;

7. la qualità del prodotto e/o del servizio fornito non è altro che il risultato dei processi aziendali attuati per realizzarlo. La qualità del prodotto è il risultato della qualità dei processi;

8. se occorre migliorare continuamente i prodotti, occorre dunque migliorare continuamente i processi aziendali;

9. per migliorare continuamente i processi aziendali occorre mobilitare la massima quantità di risorse aziendali. Presupposto del miglioramento è, dunque, il massimo coinvolgimento;

10. non è sufficiente mobilitare un elevato numero di persone per ottenere miglioramento, ma occorre organizzare questa attività e addestrare le persone a queste nuove abilità per il miglioramento richiesto.

### 2.1.1 La Qualità Totale

Quando si parla di Qualità Totale si fa sempre riferimento ai principi della competitività e del libero mercato; le norme, in questa dimensione verticale, rappresentano degli stati consolidati che permettono all'azienda di entrare nel mercato e con cui, se vorrà raggiungere l'eccellenza, dovrà continuamente confrontarsi per migliorare le proprie prestazioni.

Agire in quest'ottica dinamica e proattiva significa anzitutto:

- analizzare i processi lavorativi dell'azienda in funzione della soddisfazione del cliente al costo minimo;

- individuare e fissare ciò che viene fatto bene, affinché l'azienda possa concentrarsi sul miglioramento continuo dei processi che hanno impatto sulla qualità.

Emerge, quindi, la necessità di definire ruoli, interrelazioni e responsabilità di ogni partecipante, stabilire chi fa cosa e come lo fa, descrivere le attività svolte e documentarne i risultati, usufruendo di procedure e istruzioni scritte per ogni attività che abbia influenza sulla qualità.

Al punto 4.3 della ISO 9004:2000 sono elencati otto principi di gestione, pensati per fornire al management aziendale una guida al miglioramento delle prestazioni dell'organizzazione, apportando benefici in termini di valore, di ritorno monetario e di maggiore stabilità complessiva. I principi nascono dalla raccolta di *best practices* e dai pareri di esperti internazionali affinché i loro utilizzatori possano raggiungere il successo stabile delle organizzazioni per le quali lavorano. I concetti contenuti in questi otto principi costituiscono il fondamento su cui si basa l'intera famiglia delle norme ISO 9000 sui Sistemi di Gestione per la Qualità:

1. *Orientamento al Cliente*: le organizzazioni dipendono dai propri clienti e seguono le loro esigenze presenti e future, soddisfacendo i loro requisiti e mirando a superare le loro stesse aspettative. Benefici per l'organizzazione:

    - aumento del reddito e delle quote di mercato, grazie a una risposta più rapida e flessibile alle opportunità che il mercato stesso è in grado di offrire;

- maggiore efficacia nell'utilizzo delle risorse nel perseguire la soddisfazione dei clienti;
- maggiore fidelizzazione dei clienti, che porta non solo a una stabilità del business, ma a un suo potenziale incremento dovuto all'immagine aziendale positiva.

2. *Leadership*: la direzione generale stabilisce unità di intenti e di indirizzo dell'organizzazione. Essa crea e mantiene un ambiente interno che coinvolge pienamente il personale nel perseguimento degli obiettivi dell'organizzazione. Benefici per l'organizzazione:

   - maggiore compartecipazione e motivazione da parte di tutto il personale nel perseguire gli obiettivi e i traguardi dell'organizzazione;
   - maggiore trasparenza e coerenza nell'attuazione e successiva valutazione delle attività dell'organizzazione;
   - minor rischio che avvengano problemi di comunicazione tra i vari livelli dell'organizzazione.

3. *Coinvolgimento del personale*: le persone costituiscono l'essenza dell'organizzazione e il loro pieno coinvolgimento permette di porre le loro capacità al servizio dell'organizzazione. Benefici per l'organizzazione:

   - motivazione, rispondenza e coinvolgimento del personale nell'ambito dell'organizzazione;
   - stimolo all'innovazione e alla creatività nel raggiungimento degli obiettivi dell'organizzazione;
   - responsabilizzazione del personale per i compiti loro assegnati;
   - desiderio del personale di partecipare e contribuire al miglioramento continuo.

4. *Approccio per processi*: un risultato desiderato si ottiene con maggiore efficacia quando le relative attività e risorse sono gestite come un processo. Benefici per l'organizzazione:

   - minori costi e cicli più brevi, mediante un efficace uso delle risorse;
   - risultati migliori, coerenti e prevedibili;
   - occasioni per la messa a fuoco e la scelta delle priorità dei miglioramenti.

5. *Approccio sistemico alla gestione*: identificare, capire e gestire (come fossero un sistema) processi tra loro correlati contribuisce all'efficacia e alla efficienza dell'organizzazione. Benefici per l'organizzazione:

   - integrazione e allineamento dei processi per meglio favorire il raggiungimento dei risultati desiderati;

- capacità di mettere a fuoco i processi che più contano;
- dar fiducia alle parti interessate sulla solidità, efficacia ed efficienza dell'organizzazione.

6. *Miglioramento continuo*: il miglioramento continuo delle prestazioni complessive è un obiettivo permanente dell'organizzazione delle attività. Benefici per l'organizzazione:

   - vantaggi prestazionali attraverso migliorate potenzialità organizzative;
   - razionalizzazione delle attività di miglioramento a tutti i livelli, per perseguire gli obiettivi strategici dell'organizzazione;
   - flessibilità nel rispondere con prontezza alle opportunità che si presentano.

7. *Decisioni basate su dati di fatto*: le decisioni efficaci si basano sull'analisi di dati e di informazioni, gestite attraverso indicatori oggettivi e concordati con le persone coinvolte, sui principali parametri dei prodotti/processi. Benefici per l'organizzazione:

   - decisioni razionali, non basate esclusivamente su impressioni o intuizioni, ma su informazioni concrete;
   - maggior capacità nel dimostrare l'efficacia di precedenti decisioni, sulla base di situazioni di fatto;
   - miglior capacità di esaminare, confrontare e modificare opinioni e decisioni.

8. *Rapporti di reciproco beneficio cliente/fornitore*: il cliente e il suo fornitore sono interdipendenti e un rapporto di reciproco beneficio migliora, per entrambi, la capacità di creare valore. Benefici per l'organizzazione:

   - maggior capacità di creare valore, per entrambe le parti;
   - flessibilità e prontezza nel dare risposte congiunte al mutare del mercato o delle esigenze e aspettative dei clienti;
   - ottimizzazione di costi e risorse.

# Capitolo 3

# Le norme ISO/IEC TR 15504

## 3.1 Introduzione

La norma tecnica ISO/IEC TR 15504 fornisce un modello per la valutazione di processi software[1]. Questa valutazione esamina i processi utilizzati da un'organizzazione che opera nel settore IT per determinarne l'efficacia nel raggiungimento dell'obiettivo preposto; la valutazione può essere utilizzata con obiettivi di *Process Improvement* (miglioramento dei processi aziendali) oppure di *Capability Determination* (determinazione delle capacità dell'azienda in funzione dell'analisi delle prestazioni dei processi). La valutazione caratterizza l'attività corrente all'interno dell'unità organizzativa in termini di capacità del processo selezionato. I risultati ottenuti dalla valutazione possono essere utilizzati per attività di miglioramento dei processi oppure di determinazione delle capacità di processo tramite l'analisi dei risultati stessi nel contesto delle necessità di business dell'azienda identificando punti di forza, debolezza e rischi inerenti al processo.

La norma tecnica ISO/IEC TR 15504 è suddivisa in nove parti, unificabili sotto il titolo `Information Technology - Software process assessment´

- Parte 1: *Concetti e guida introduttiva.* Presenta una guida alla normativa.
- Parte 2: *Modello di riferimento per i processi e le capacità di processo.* Descrive le caratteristiche che un modello di riferimento deve avere al fine di ottenere una buona valutazione.
- Parte 3: *Esecuzione di una valutazione.* Illustra le fasi della valutazione ed i documenti da produrre.

[1] Quando si realizza un prodotto o un sistema software è importante svolgere una serie di passi, una sorta di percorso che aiuti ad ottenere risultati di alta qualità in un tempo prefissato. Tale percorso è chiamato *processo software.* Il processo software è composto da alcune attività che rappresentano un insieme di compiti da svolgere per sviluppare un software: attività portanti, da svolgere necessariamente, e attività ausiliarie che possono aumentare la qualità di un software da produrre.

- Parte 4: *Guida all'esecuzione di una valutazione.* Descrive in dettaglio tutte le diverse fasi della valutazione per guidare i valutatori.
- Parte 5: *Modello di valutazione e guida sugli indicatori.* Contiene un esempio di modello compatibile.
- Parte 6: *Guida alla competenza degli esaminatori.* Presenta una guida sulle competenze ed abilità necessarie ad un valutatore per dirigere in modo corretto la valutazione.
- Parte 7: *Guida all'uso per il miglioramento dei processi.*
- Parte 8: *Guida all'uso per la determinazione delle capacità di processo del fornitore.*
- Parte 9: *Vocabolario.* Include un dizionario di riferimento.

In Figura 3.1 possiamo vedere la distinzione delle varie parti e la loro interazione.

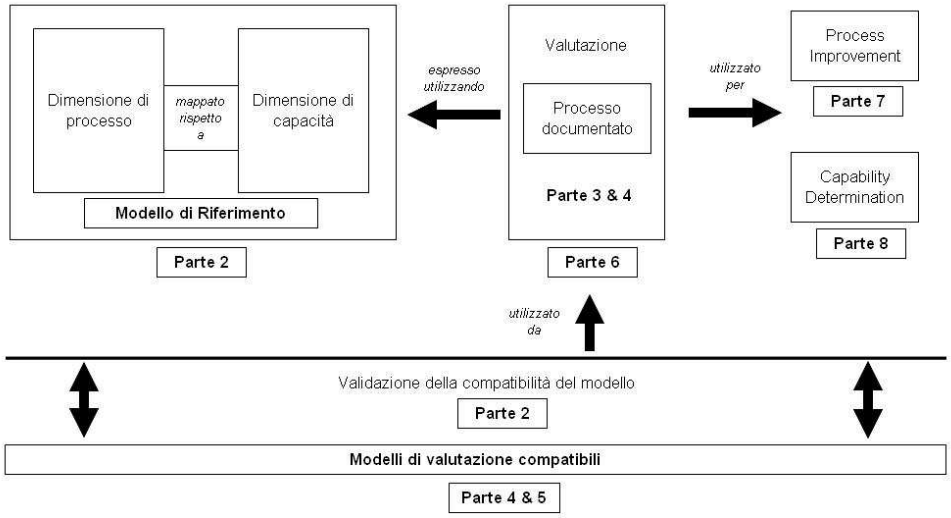


**Figura 3.1.** Parti della norma ISO/IEC TR 15504 e loro interazione.

Lo schema per la conduzione di una valutazione è progettato per supportare il raggiungimento di risultati attendibili. Esso include un'architettura per il processo di valutazione e per il processo di presentazione della valutazione. Lo schema inoltre fornisce una guida per la conduzione della valutazione sia nel contesto di Process Improvement che in quello di Capability Determination.

L'inizio formale del processo di valutazione coincide con l'avvio deciso dalla dirigenza (o sponsor della valutazione). L'input della valutazione definisce

l'obiettivo (perchè la valutazione viene effettuata), l'ambito ed i limiti imposti e le responsabilità per la gestione della valutazione. Il processo di valutazione è effettuato tramite il confronto tra processi precedentemente selezionati e processi selezionati da un modello scelto per la valutazione (modello che verrà illustrato nel capitolo 5). Questo modello deve essere compatibile con quello di riferimento definito nella Parte 2 della norma tecnica. Il modello di riferimento consiste in uno schema bidimensionale dove una dimensione è rappresentata dal set di processi, mentre l'altra consiste nel set di attributi di processo. Gli attributi sono applicabili ad ogni processo, essi sono raggruppati in livelli di capacità che possono essere utilizzati per determinare la capacità del processo. L'output della valutazione consiste in un set di profili di processi ed opzionalmente anche in una valutazione del livello di capacità per i processi in esame. Il processo di valutazione si basa in almeno cinque attività: pianificazione, collezione dati, validazione dei dati, valutazione dei processi e reporting dei voti. L'intero processo deve essere documentato, inoltre il valutatore deve lasciare traccia di indicatori oggettivi di performance utilizzati durante la valutazione. La tracciatura e la conservazione dei risultati della valutazione sono necessarie al fine di massimizzare la ripetitività, l'affidabilità e la consistenza.

## 3.2 Esecuzione della valutazione

La valutazione dei processi viene effettuata per comprendere le prestazioni dei processi correnti di un'unità organizzativa. La valutazione tiene conto di tutti i processi software utilizzati da un'azienda. In alcune circostanze potrebbe essere utile comparare i risultati di diverse valutazioni o di valutazioni appartenenti a due o più differenti unità organizzative (per esempio comparare le valutazioni svolte sulle organizzazioni candidate a diventare fornitori per effettuare una selezione).

La valutazione può essere gestita in accordo a due differenti approcci:

- **Auto-valutazione** L'autovalutazione è eseguita da un'organizzazione per valutare le capacità dei propri processi software. Lo sponsor della valutazione è tipicamente interno all'organizzazione.

- **Valutazione indipendente** Questo tipo di valutazione è condotto da valutatori indipendenti dalle unità organizzative sotto esame. Lo sponsor di una valutazione indipendente può essere esterno all'unità organizzativa sotto esame, come, ad esempio, un acquirente il cui obiettivo è di ottenere una valutazione indipendente. Il grado di indipendenza può variare in accordo con gli obiettivi e le circostanze della valutazione.

La valutazione dovrebbe essere eseguita in accordo a un processo documentato capace di garantire obiettivi prestabiliti. Gli elementi chiave del processo di valutazione sono un elevato livello di documentazione, l'utilizzo dii modelli

di valutazione compatibili alla norma e l'uso di strumenti e tool di supporto. In ogni valutazione sarà necessario raccogliere, documentare, salvare, riordinare, analizzare recuperare e presentare le informazioni. In generale, un processo di valutazione documentato è supportato da vari strumenti e tool per la raccolta, l'analisi e la presentazione delle informazioni. Si dovrebbero dichiarare in anticipo gli obiettivi per cui questi strumenti sono utilizzati, in modo da aiutare i valutatori a compiere una valutazione in una modalità consistente e ripetibile.

Per ottenere un buon risultato nella valutazione, i seguenti fattori sono essenziali:

- **Impegno**
  Lo sponsor si dovrebbe impegnare, al fine di raggiungere gli obiettivi prefissati, a fornire l'autorità necessaria per lo svolgimento della valutazione. Questo impegno richiede che le risorse, di tempo e personale, necessarie per l'attività di autovalutazione siano effettivamente rese disponibili per lo svolgimento della valutazione.

- **Motivazione**
  L'atteggiamento del management, e le modalità del processo di valutazione in accordo al quale le informazioni sono raccolte, hanno un'influenza significativa sull'esito della valutazione. Il management aziendale deve motivare i partecipanti ad essere aperti e costruttivi. La valutazione si focalizza nei processi e non nelle performance ottenute dai membri delle unità organizzative. L'intento è quello di sviluppare il processo al fine di renderlo più efficace in accordo agli obiettivi di business predefiniti e non certo quello di ricercare manchevolezze nei singoli individui.

- **Confidenzialità**
  Il rispetto della confidenzialità sia delle sorgenti di informazione sia della documentazione raccolta durante la valutazione è essenziale al fine di garantire l'integrità e correttezza delle informazioni. Se vengono utilizzate tecniche di discussione diretta, si dovrebbe fare in modo che i partecipanti non si sentano sotto esame o minacciati nel diritto alla privacy.

- **Rilevanza**
  I membri delle unità organizzative dovrebbero essere credere che la valutazione avrà come effetti alcuni benefici che li riguarderanno in maniera diretta o indiretta.

- **Credibilità**
  Lo sponsor, la direzione e lo staff dell'unità organizzativa dovrebbero essere convinti che la valutazione possa fornire dei risultati obiettivi e rappresentativi per la quota parte di lavoro corrente o responsabilità di loro competenza. Inoltre, è importante che tutte le parti siano confidenti che

i valutatori abbiano adeguata esperienza nella valutazione, che siano imparziali e che abbiano una conoscenza adeguata delle unità organizzative e del loro lavoro.

### 3.2.1 Input della valutazione

Prima di far avanzare il processo di valutazione dell'organizzazione, è necessario aver identificato tutti i parametri che costituiscono l'input del processo stesso. Tali parametri devono comprendere i seguenti aspetti:

- **L'obiettivo della valutazione e di business**
  Differenti tipi di valutazioni (Process Improvement o Capability Determination) hanno differenti obiettivi. L'obiettivo deve tenere in debita considerazione le necessità di business.

- **L'ambito della valutazione**
  In genere la valutazione tiene conto di tutti i processi software utilizzati da un'azienda. Tuttavia, per ragioni di praticità, l'ambito può essere ridotto a un insieme più ristretto di processi, con conseguente effetto negativo sull'accuratezza della valutazione stessa. La selezione delle unità organizzative dovrebbe riflettere ciò che lo sponsor vuole ottenere dai risultati della valutazione. Gli elementi da considerare ai fini di tale esclusione sono quindi le relazioni tra gli obiettivi della valutazione e l'abilità di fornire giudizi, l'attuale livello di prestazioni di ciascun processo e i vincoli nella durata della valutazione.

- **I vincoli della valutazione**
  Il successo della valutazione può risentire della scarsità di risorse messe a disposizione. Si deve inoltre cercare di minimizzare l'interruzione del lavoro delle unità organizzative sotto indagine. Il processo e l'ambito possono essere aggiustati in base alla quantità di tempo a disposizione, a questo scopo potrebbe essere necessario operare alcune esclusioni tenendo presente gli aspetti già evidenziati.

- **I modelli utilizzati**
  Per facilità di applicazione sarebbe preferibile utilizzare un solo modello compatibile alla norma come elemento di confronto.

- **L'identità dei valutatori**
  La conoscenza e l'esperienza dei valutatori migliora la qualità dei risultati della valutazione. La partecipazione di valutatori interni all'unità organizzativa al team di valutazione può aiutare a fornire informazioni sul contesto dei processi analizzati e motiva il personale.

- **Le competenze dei valutatori**
  Nella sesta parte della norma è fornita una guida per lo sponsor riguardo

alle competenze dei valutatori. Il processo di valutazione deve fornire criteri specifici per definire preventivamente i requisiti ai quali deve soddisfare un valutatore per poter essere qualificato.

- **L'identità degli oggetti della valutazione**
  La selezione degli oggetti della valutazione dovrebbe essere rappresentativa delle unità organizzative da analizzare. Se i partecipanti sono rappresentativi dell'unità organizzativa i risultati rispecchianole prestazioni di ogni processo sotto indagine. Diversamente, i risultati della valutazione potrebbero perdere di attendibilità.

- **Informazioni addizionali**
  Tutte le informazioni in supporto al contesto del processo, come opportunità di miglioramento o rischi nell'acquisizione dei dati, devono essere documentate.

### 3.2.2 Output della valutazione

Le informazioni relative ai risultati della valutazione dovrebbero essere compilate ed incluse nella documentazione per la revisione da parte degli sponsor. Come minimo, l'output della valutazione dovrebbe contenere:

- la data della valutazione;
- gli input della valutazione;
- gli identificatori dell'evidenza oggettiva raccolta;
- l'approccio utilizzato per la valutazione;
- il set dei profili di processo risultante dalla valutazione;
- l'identificazione di qualsiasi informazione addizionale collezionata durante la valutazione ed identificata negli input della valutazione in supporto al miglioramento dei processi o alla determinazione delle performance dei processi.

### 3.2.3 Fasi della valutazione

Come già detto, un processo di valutazione documentato sostiene la ripetitività dell'approccio di valutazione e fornisce le basi per il miglioramento continuo. Le attività sequenziali e le procedure che lo contraddistinguono sono le seguenti:

**Fondamentali** Il processo di valutazione dovrebbe descrivere come vengono raccolte, validate, approvate e documentate tutte le informazioni richieste per l'input della valutazione. Il processo di valutazione dovrebbe fornire supporto per la registrazione o il trasferimento degli input in una modalità adatta ad

essere parte dell'output. Il processo di valutazione dovrebbe fornire una guida su come:

- ottenere l'impegno dello sponsor;
- scegliere il modello di riferimento per la valutazione;
- definire le proprietà degli output;
- pianificare la valutazione;
- gestire la confidenzialità;
- classificare il contesto di ciascun processo valutato;
- verificare il rispetto dei requisiti per la valutazione.

Il processo di valutazione dovrebbe definire dei meccanismi per:

- permettere di eseguire la valutazione efficientemente entro i limiti predefiniti, o come i limiti e/o l'ambito possono essere rinegoziati e approvati se non è possibile eseguire la valutazione;
- sostenere la raccolta di qualsiasi altra informazione richiesta dallo sponsor.
- permettere allo sponsor di assicurare che il valutatore nominato acquisisca o abbia le competenze per svolgere la valutazione;
- validare i valutatori;
- definire le ulteriori regole che si rendano necessarie in corso d'opera, le responsabilità all'interno della valutazione e le competenze richieste per ogni ruolo;
- assicurare che la valutazione sia conforme con i requisiti forniti nelle Parti 2 e 3 della normativa (Modello di riferimento ed Esecuzione della valutazione);
- definire come la conformità (rispetto al modello adottato) sia ottenuta e per validarla.

**Raccolta dati** Il processo di valutazione dovrebbe fornire una guida sul meccanismo di raccolta dati così come sulle tecniche di intervista e sugli strumenti di revisione dei documenti. Inoltre dovrebbe fornire una guida per identificare come i processi delle unità organizzative sono mappati nei processi definiti all'interno del modello fornito dalla norma.

**Validazione dati** Il processo di valutazione dovrebbe fornire una guida sulla validazione delle informazioni la quale includa, come minimo, pareri da parte di sorgenti indipendenti, informazioni su come utilizzare i risultati delle valutazioni precedenti, come effettuare le sessioni di feedback per validare le informazioni raccolte.

**Valutazione dei processi** Il processo di valutazione dovrebbe fornire un meccanismo per assegnare le valutazioni ai componenti predefiniti nel modello selezionato. Quando questi componenti sono differenti da quelli predefiniti nella parte 2 della normativa, il processo di valutazione dovrebbe fornire una guida su come tradurre i valori raccolti in giudizi. Il processo di valutazione dovrebbe, inoltre, definire un meccanismo per validare le votazioni assegnate ai processi.

**Documentazione** Il processo di valutazione permette di documentare le informazioni raccolte rispettando alcuni requisiti. In particolare dovrebbe:

- fornire un meccanismo per la documentazione delle informazioni e giudizi associati agli indicatori definiti nel modello selezionato per la valutazione;
- specificare il formato con cui verranno presentati i risultati allo sponsor, alle persone valutate e ogni altra parte interessata;
- definire come la documentazione sarà conservata; ad esempio in forma cartacea o elettronica a seconda delle circostanze e dei tools di supporto utilizzati;
- definire le modalità di conservazione dei dati da parte dello sponsor, dei valutatori, dell'organizzazione valutata, o da terze parti in accordo ai requisiti di confidenzialità identificati nell'input della valutazione;
- fornire un meccanismo di conservazione e accesso ai dati raccolti in accordo alle modalità di cui al punto precedente.

## 3.3 Il team di valutazione

Il ruolo del team di valutazione è quello di valutare i processi software di un'organizzazione in modo costruttivo ed obiettivo come descritto nella norma ISO/IEC TR 15504-4. La valutazione dovrebbe focalizzarsi su ciascun processo e non sulle persone che realizzano i processi. Il ruolo del valutatore assume connotazioni differenti a seconda del caso di autovalutazione o di valutazione esterna. La tabella 3.1 evidenzia le corrispondenti differenze in forma comparativa. La figura 3.2 mostra come i valutatori debbano dimostrare le loro competenze per svolgere le valutazioni, da cosa conseguono le competenze e come la conoscenza, le abilità e le qualità personali siano ottenute dalla combinazione di istruzione, addestramento ed esperienza.

| **Autovalutazione** | **Valutazione indipendente (esterna)** |
|---|---|
| È orientato alle persone e ai compiti | È orientato ai compiti |
| Dirige la valutazione | Controlla la valutazione |
| Accorda una classifica | Consegna una classifica |
| Promuove la discussione | Regola la discussione |
| Lavora con i progetti | Lavora in modo separato dai progetti |
| Utilizza gli obiettivi di business dell'unità organizzativa | Potrebbe essere indifferente agli obiettivi di business dell'unità organizzativa |
| Influenza attraverso i risultati ottenuti, le relazioni stabiliti e la competenza | Influenza attraverso la posizione e la competenza |
| Ricerca responsabilità | Determina l'adeguatezza dei processi |
| È simile ad un agente di cambio | È simile ad un revisore contabile |

**Tabella 3.1.** Ruolo del valutatore a seconda del tipo di valutazione da questi svolte.

Un *valutatore provvisorio* è una persona che ha raggiunto il livello richiesto di istruzione, addestramento ed esperienza, ma che non ha ancora partecipato ad una valutazione condotta in accorto alle prescrizioni della ISO/IEC TR 15504. Un valutatore provvisorio dovrebbe essere addestrato e acquisire esperienza relativamente al processo software in aggiunta alla valutazione del processo o alla valutazione della qualità del software. Egli dovrebbe ricevere un addestramento tale che soddisfi la norma ISO/IEC TR 15504 e possedere un evidente livello di istruzione (intesa come cultura generale e specifica rispetto al software). Livelli soddisfacenti di istruzione potrebbero essere raggiunti tramite:

- corsi offerti da college o università;
- corsi professionali organizzati da enti locali o internazionali riconosciuti;
- corsi sponsorizzati dai venditori;
- corsi sponsorizzati dal datore di lavoro.

Livelli soddisfacenti di addestramento potrebbero essere conseguiti per mezzo di:

- addestramento fornito da enti locali o internazionali riconosciuti,
- addestramento fornito da venditori e addestramenti basati sulla guida fornita da questa parte della ISO/IEC TR 15504.

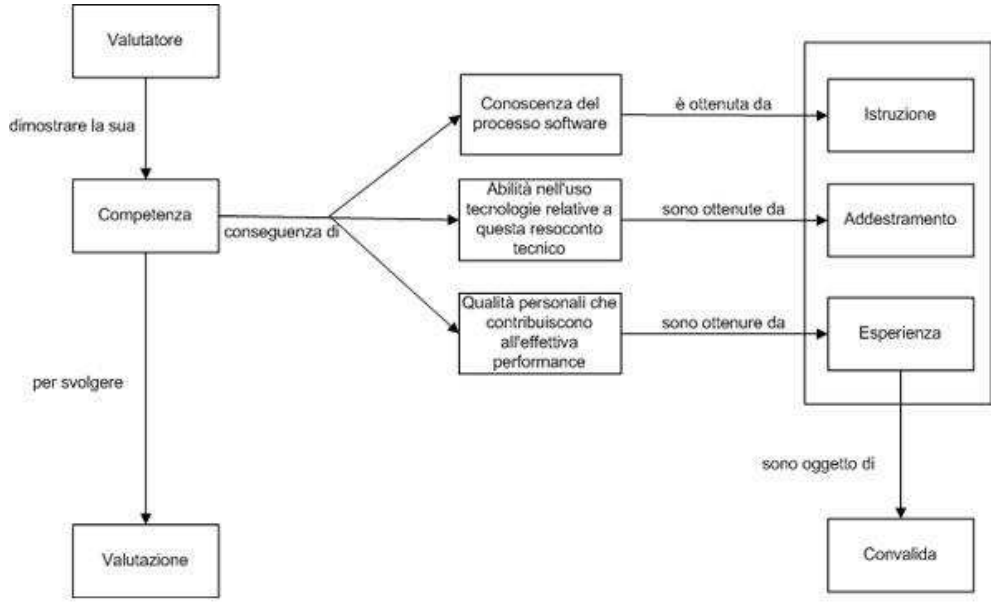


**Figura 3.2.** Competenze del valutatore.

Livelli soddisfacenti di esperienza potrebbero acquisiti tramite:

- esperienza diretta in aeree specialistiche come l'ingegneria del software, lo sviluppo - mantenimento del software o la qualità del software;
- esperienza manageriale di sorveglianza in aeree specialistiche come l'ingegneria del software, lo sviluppo - mantenimento del software o la qualità del software.

Un *valutatore competente*, deve aver partecipato ad una valutazione condotta in accordo alla norma ISO/IEC TR 15504. I valutatori competenti dovrebbero conservare documentazione delle attività professionali in corso per dimostrare l'evoluzione delle competenze riguardanti le abilità, la conoscenza e l'addestramento.

Un valutatore dovrebbe possedere familiarità con lo sviluppo di software e il suo mantenimento e dovrebbe essere in grado di dimostrare competenza in almeno una delle categorie del modello del processo descritte nell'ISO/IEC TR 1550-2. In aggiunta, un valutatore dovrebbe mostrare di aver compreso le attività richieste per il supporto del processo software, i corrispondenti metodi ed essere in grado di valutare correttamente quando e come dovrebbero essere utilizzati. In sintesi, un valutatore dovrebbe avere, quindi, familiarità con una rilevante serie di standard di ingegneria del software.

Per mantenere le loro competenze, i valutatori dovrebbero mantenere aggior-

nata la loro conoscenza impegnandosi in attività professionali in aggiunta allo svolgimento delle valutazioni secondo la norma ISO/IEC 15504.

I valutatori, inoltre, dovrebbero possedere le seguenti qualità personali:

- capacità di comunicazione in modo efficace sia in forma scritta, sia orale;
- diplomazia;
- Discrezione;
- capacità di giudizio e di direzione;
- integrità[2].

I valutatori dovrebbero mantenere prove documentate della loro istruzione conservando certificati e programmi dei corsi di formazione seguiti. I livelli di istruzione seguenti sono normalmente considerati coerenti con la categoria di istruzione generale e quella riguardante il software.

- **Istruzione generale:** laurea o titolo equivalente in qualsiasi disciplina fornito da ente universitario.
- **Istruzione riguardante il software:** laurea o titolo equivalente in Informatica, Ingegneria del Software o consimili.

I valutatori dovrebbero essere competenti in tutti i processi della categoria *ingegnerizzazione del software*. L'addestramento nella gestione dei progetti o la direzione tecnica forniscono un background nelle categorie riguardanti i processi *Customer Supplier* (CUS) e i processi *Organizational* (ORG). I valutatori non necessitano un addestramento specifico in ogni processo previsto dalla norma, ma dovrebbero avere dimestichezza generale con i corrispondenti temi. Invece, i valutatori dovrebbero essere ben addestrati in almeno uno dei processi di queste due categorie.

## 3.4 Come realizzare il processo di miglioramento

La Parte 7 della norma ISO/IEC TR 15504 fornisce una guida sull'utilizzo della valutazione del processo software come elemento per realizzare il miglioramento continuo del software. La guida non fa riferimento ad una ben specifica struttura organizzativa, a una filosofia di management, a un ciclo di vita del software oppure a particolari metodi per lo sviluppo del software. I concetti ed i principi presentati nella norma sono, invece, adatti a tutte le diverse tipologie di aziende, domini di applicazioni e dimensioni dell'organizzazione. Un'organizzazione potrebbe scegliere tutti oppure solo alcuni sottoinsiemi dei processi di software

[2]Con il termine "integrità" si intende l'assenza di conflitti di interesse nello svolgimento della valutazione.

dal modello di riferimento per la valutazione e il miglioramento in accordo a specifiche esigenze.

Gli obiettivi di un'organizzazione dovrebbero comprendere la soddisfazione dell'utente finale, l'aumento della competitività e l'incremento del valore del software o dei sistemi informativi realizzati. Per le organizzazioni in cui il software è un elemento centrale, questi obiettivi di management diventano fondanti rispetto al processo di miglioramento del software in tutta l'organizzazione, con gli obiettivi di aumentare la qualità del software, di diminuire il costo di sviluppo e mantenimento, di accorciare il time-to-market ed incrementare il livello di controllo dei prodotti e processi software. I miglioramenti al processo software possono essere avviati a qualsiasi livello dell'organizzazione. Tuttavia, è richiesto alla direzione di promuovere per prima e sostenere il cambiamento e di fornire le necessarie risorse.

Il contesto per il processo di miglioramento del software e gli elementi principali in ingresso/uscita sono schematizzati nella fig. 3.3:

- le necessità e gli obiettivi di business dell'organizzazione che sono il principale stimolo per il processo di miglioramento;
- le normative industriali e i benchmark forniscono le informazioni di riferimento per la pianificazione dei miglioramenti;
- i miglioramenti nel processo software dell'organizzazione sono i risultati.

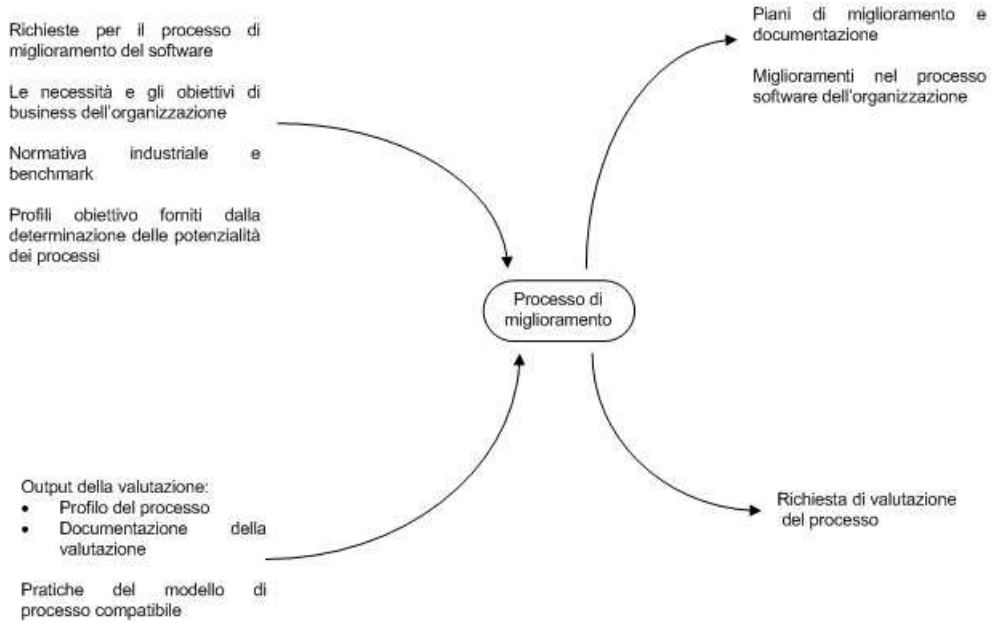


**Figura 3.3.** Contesto del processo di miglioramento.

Il processo di miglioramento tiene conto anche di altri aspetti dell'ISO/IEC 15504:

- il processo di valutazione è realizzato per stabilire le capacità correnti;
- i risultati della valutazione consistono in profili di processi in informazioni contestuali raccolte nella documentazione della valutazione;
- il modello di riferimento (come quello specificato nell'ISO/IEC TR 15504-2) è usato come un framework per definire i processi da migliorare, impostare le priorità e identificare le azioni per il miglioramento;
- i piani di miglioramento e le registrazioni possono aiutare nello stabilire le performance del processo.

### 3.4.1 Linee guida per il processo di miglioramento del software

La figura 3.4 mostra una articolazione del processo di miglioramento del software da seguire come guida alla sua definizione. I benefici di tale processo sono permanenti quando un'organizzazione svolge attività di miglioramento con impegno e disciplina servendosi anche di rigorose raccolte dati.

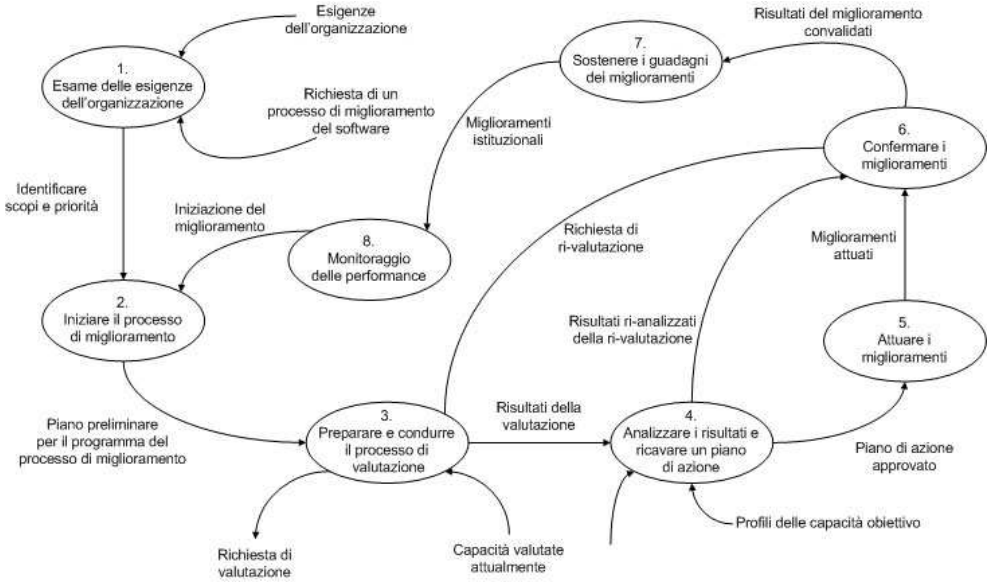


**Figura 3.4.** Linee guida per il processo di miglioramento del software.

Un possibile programma di un processo di miglioramento inizia con il riconoscimento degli obiettivi di business dell'organizzazione. A partire da tale analisi delle esigenze dell'organizzazione, possono essere identificati e descritti gli obiettivi del processo di miglioramento in termini di qualità, time-to-market,

costi, soddisfazione dell'utente finale, valore dei risultati attesi e rischi. Quindi, si procede ad identificare le priorità di ciascun obiettivo. Gli obiettivi trovati in tal modo dirigono la scelta dei processi da valutare, forniscono la definizione dei valori di miglioramento, obiettivo di ciascun processo e finalmente permettono di formulare le azioni di miglioramento più efficaci.

Questo processo è, quindi, a tutti gli effetti un progetto pianificato, attuato e gestito attentamente. All'inizio si deve, quindi, produrre un piano che in seguito va utilizzato per monitorare i progressi. Nel piano dovrebbe essere inclusa una valutazione della storia pregressa e dello stato corrente, per quanto possibile in una terminologia specifica e quantitativa. Il piano, poi, dovrebbe contenere una descrizione delle fasi e delle azioni previste. È importante anche identificare le risorse necessarie. Inoltre, devono essere fissati a priori metriche e punti di misura. Responsabilità, rischi o aspetti critici di attuazione dovrebbero essere identificati e documentati nel piano.

La preparazione della valutazione inizia con l'identificazione di uno *sponsor*. Lo *sponsor* è un senior manager, impiegato nel programma del processo, il quale richiede che la valutazione venga svolta e mette a disposizione le risorse per eseguirla. Lo sponsor garantisce che gli ingressi del processo di valutazione (scopi, ambiti, vincoli e responsabilità) siano adeguatamente definiti in modo da andare incontro alla necessità del programma di miglioramento. La responsabilità di garantire che una valutazione venga svolta in accordo alla norma ISO/IEC TR 15504 è rivestita dal valutatore competente che conduce il team di valutazione. Un fattore chiave è selezionare un valutatore che sia credibile agli occhi dello staff di management e all'organizzazione. A seconda delle circostanze locali, un valutatore competente scelto all'esterno dell'organizzazione potrebbe apparire più credibile invece di tenere conto di molti punti di vista indipendenti. Lo scopo globale della valutazione è fornire informazioni sulle capacità del processo dell'organizzazione sottoforma di risultati. L'ambito della valutazione definisce i limiti per la valutazione, sia organizzativi sia in termini di processi da coinvolgere. Più ampio è l'ambito di una valutazione, maggiore sarà lo sforzo valutativo necessario per arrivare a risultati rappresentativi. Quindi, lo sponsor potrebbe voler limitare l'ambito a quei processi per i quali è atteso il più grande miglioramento potenziale. In definitiva, tuttavia, i processi organizzativi necessitano di essere convertiti in processi appartenenti al modello compatibile dei processi, per consentire al team di condurre la valutazione. Se un processo organizzativo non può essere convertito in uno dei processi del modello, dovrebbe essere definito come un processo esteso. Lo sponsor potrebbe voler restringere la libertà di scelta del team di valutazione nel definire la strategia della valutazione, nel selezionare gli individui per l'intervista, e in che modo le informazioni possano essere usate.

Le informazioni raccolte durante la valutazione, in particolare il livello delle capacità e la classifica degli attributi dei processi, sono analizzate alla luce delle esigenze dell'organizzazione di:

- identificare le aree da migliorare;

- impostare scopi qualitativi per il processo software e obiettivi qualitativi di miglioramento;

- ricavare un piano di azione, e integrarlo con il piano del programma del processo di miglioramento.

Le aree di miglioramento dovrebbero essere identificate ed a loro assegnate una priorità sulla base di alcuni fattori mostrati nella figura 3.5:

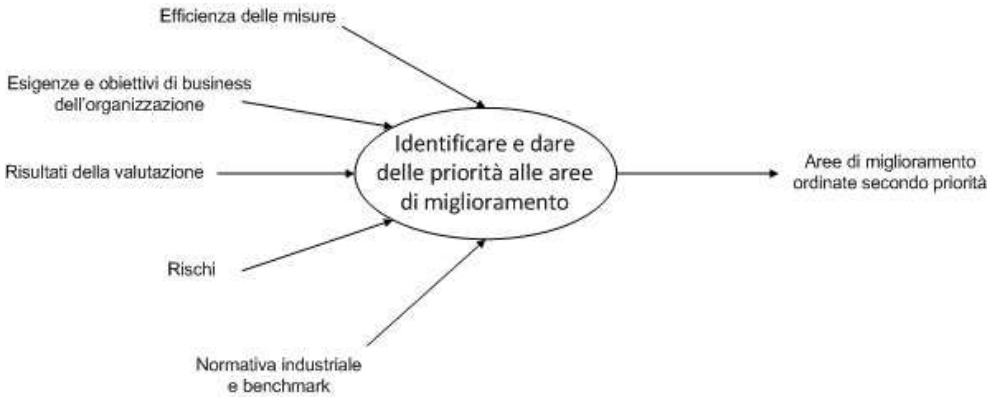


**Figura 3.5.** Aree di miglioramento.

Il piano aggiornato del programma del processo di miglioramento è sviluppato allo scopo di migliorare il processo software dell'organizzazione. Lo sviluppo può essere semplice o complesso a seconda del contesto delle azioni incluse nel piano aggiornato e delle caratteristiche dell'organizzazione.

Quando il progetto per il processo di miglioramento è stato completato, l'organizzazione dovrebbe:

- confermare che gli obiettivi siano stati raggiunti e i benefici attesi siano stati ottenuti;

- confermare che la cultura organizzativa desiderata sia stata stabilita;

- rivalutare i rischi associati al processo di miglioramento;

- rivalutare costi e benefici.

Dopo che i miglioramenti sono confermati, il processo software necessita di essere portato ad un nuovo livello di performance. Se un processo migliorato è stato pilotato in un'area ristretta oppure in uno specifico progetto o gruppo di progetti, esso dovrebbe essere sviluppato attraverso tutte le aree o progetti

quando è possibile. Questo impiego dovrebbe essere pianificato in modo opportuno e assegnando le necessarie risorse. Il piano dovrebbe essere documentato come parte del piano del progetto per il miglioramento del processo.

Le performance del processo software di un'organizzazione dovrebbero essere continuamente monitorate, e il nuovo progetto per il processo di miglioramento dovrebbe essere selezionato come parte del programma di miglioramento continuo.

# Capitolo 4

# Il caso di studio

In questo capitolo viene presentata una breve descrizione del profilo aziendale dell'azienda per cui è stata eseguita la valutazione. Nei paragrafi successivi, inoltre, vengono descritti i processi aziendali identificati attraverso interviste effettuate al personale.

## 4.1 Profilo aziendale

Mida Solutions è un'azienda italiana che progetta, realizza e commercializza soluzioni software per Servizi Voce e Applicazioni a Valore Aggiunto, per il mondo della telefonia IP e tradizionale, al fine di dare un contributo significativo al miglioramento delle funzionalità telefoniche. L'azienda è stata fondata nel 2004, con la missione di fornire tecnologie innovative a valore aggiunto per le telecomunicazioni, puntando sia sul mercato nazionale che su quello internazionale. Mida Solutions è un'azienda che ha sede a Padova e che collabora strettamente con l'Università locale; il focus dell'azienda è sempre stato su tecnologie innovative e sviluppo delle risorse umane. I prodotti di Mida Solutions sono basati su tecnologie d'avanguardia nel mondo della comunicazione, come protocolli SIP e H323, Data Base ad alte prestazioni, motori di reportistica professionali, tool di RiconoscimentoVocale.

## 4.2 Modello di descrizione dei processi aziendali

Il primo passo per l'informatizzazione del sistema di gestione della qualità è la definizione di un modello descrittivo per la rappresentazione dei processi aziendali. In questa prima fase devono essere individuate tutte le esigenze aziendali con il fine di isolare le possibili soluzioni; esse devono essere caratterizzate da espressività descrittiva, presenza di tool per la gestione, semplicità nel disegno e nella lettura degli schemi. Per raggiungere un buon compromesso tra flessibilità ed espressività, analisti (coloro i quali studiano e propongono le possibili soluzioni) e utenti finali (chi realmente utilizzerà in futuro la soluzione adottata) devono

collaborare tra di loro instaurando un canale di comunicazione; esso permette lo scambio di informazioni finalizzate alla ricerca e alla definizione del modello descrittivo. Solamente con un continuo scambio di dati, tra le due parti, si può isolare la soluzione che più si adatta alle richieste iniziali.

In questa sezione vengono riportate le problematiche incontrate durante il lavoro di modellizzazione di una specifica azienda (Mida Solutions); le quali, dopo una prima fase di generalizzazione, sono state risolte con l'obiettivo di definire un modello descrittivo applicabile ad una qualsiasi realtà. La volontà di realizzare un modello descrittivo generico, ma completo, ha richiesto un notevole sforzo, il quale, però, ha permesso di creare uno strumento flessibile e di facile comprensione, capace di rappresentare processi aziendali semplicemente concatenando elementi basilari.

### 4.2.1 Proprietà del modello

Uno dei principali problemi che si possono incontrare nell'analisi di un'azienda riguarda l'identificazione e la descrizione dei processi aziendali. La difficoltà risiede nel riuscire ad esprimere tutti i particolari che caratterizzano un processo, senza rendere la rappresentazione poco leggibile e conseguentemente difficile da gestire. È quindi molto importante individuare gli elementi che devono essere presenti nel modello tenendo presente i seguenti fattori:

- grado di conoscenza e preparazione degli utenti finali della descrizione. È infatti importante individuare immediatamente chi dovrà leggere e comprendere il diagramma del processo. Nella pratica si ha spesso la necessità di offrire un buon grado di comprensione a tutte le figure aziendali (dal responsabile per la qualità agli impiegati). Per questo motivo la descrizione deve essere semplice ed intuitiva;

- grado di dettaglio che si vuole ottenere dalla descrizione. Il processo può essere descritto utilizzando vari gradi di dettaglio. Questo è spesso necessario nelle situazioni in cui l'azienda ha dimensioni notevoli per cui la gerarchia aziendale risulta molto profonda, ed è quindi estremamente probabile, che vari livelli della gerarchia stessa abbiano bisogno di diversi gradi di dettaglio. Questo non è semplicemente un fatto di comodità, ma deriva anche dal livello di preparazione ed interesse che le figure possiedono. In alternativa si può richiedere che la descrizione risulti allo stesso tempo semplice ed espressiva qualora non si prevedano diversi livelli di astrazione;

- grado di formalismo che si vuole attribuire alla descrizione. Con il temine formalismo vanno intese tutte le regole (grafiche e non) per la rappresentazione di un processo. L'adozione di una rappresentazione puramente grafica permette sicuramente di poter conferire alle descrizioni un elevato grado di espressività, ma spesso non permette di definire delle regole

precise e formali che governino la rappresentazione stessa. Viceversa l'uso di una rappresentazione estremamente formale potrebbe dar luogo a rappresentazione di difficile comprensione e di ancora più difficile utilizzo. Nell'effettuare l'analisi del formalismo è importante capire quanta "rigidità" imporre alle regole; ciò può essere stabilito solamente mantenendo, come già detto, un continuo contatto con gli utenti finali.

Nel caso specifico, l'individuazione di un formalismo è ulteriormente complicata dalla necessità di utilizzarlo per due scopi apparentemente ortogonali:

1. offrire una descrizione adatta alla "lettura umana" adottando, quindi, una rappresentazione che sia di facile comprensione per i vari attori coinvolti;

2. offrire una descrizione adatta all'automatizzazione scegliendo, quindi, una rappresentazione che sia di facile comprensione e gestione da parte di un software.

Si può quindi capire che la fase di individuazione del formalismo ha rappresentato uno tra gli elementi centrali nello sviluppo di questo lavoro. Si vuole, inoltre, sottolineare che tale scelta non è stata unicamente sostenuta da analisi e considerazioni di tipo "teorico"; infatti, parallelamente, si è svolta una ricerca finalizzata all'individuazione di tool software capaci di agevolare l'utilizzo di tali formalismi. Ignorare questo aspetto significa accettare il fatto che, una volta individuata la rappresentazione da utilizzare, si deve provvedere alla creazione di un software ad hoc per la manipolazione.

### 4.2.2 L'analisi di Mida Solutions

Il lavoro di analisi di una realtà quale Mida Solutions ci ha permesso di verificare sul campo che le richieste avanzate nei paragrafi precedenti sono più che lecite se si è spinti dalla volontà di fornire uno strumento che permetta l'informatizzazione del sistema di qualità senza appesantire ulteriormente il carico di lavoro aziendale.

La prima fase dell'analisi si è concentrata sull'instaurazione di una canale di comunicazione con l'azienda; mediante il quale, grazie a interviste frontali con le varie figure dell'azienda, si sono delineati i processi e gli elementi che li caratterizzano. Durante questo primo scambio di idee si sono impiegati, per facilità di rappresentazione, diagrammi di control flow; essi permettono di descrivere gli elementi che particolarizzano i processi con estrema facilità, inoltre permettono di rappresentare i vincoli di precedenza tra azioni esprimendo implicitamente il punto di inizio e di fine del processo. Le interviste, oltre a far emergere la struttura aziendale, si sono dimostrate fondamentali per tracciare le proprietà del modello; esse hanno fatto si che si potesse delineare la preparazione degli utenti finali, il livello di granularità dei dettagli e il grado di formalismo da attribuire alla descrizione.

Le informazioni inerenti ai dati prodotti e alla loro gestione sono state raccolte con cura senza dimenticare di riportare i ruoli aziendali che operano su di essi e con quali permessi. Collezionate tutte le informazioni necessarie, l'analisi si è focalizzata sulla ricerca di uno strumento già esistente in commercio che potesse rappresentare la realtà studiata. Si sono, quindi, estratti dalle informazioni raccolte gli elementi che potessero costituire la base del modello, così da determinare i criteri di ricerca. Nel tentativo di delineare gli elementi base si è dedicato del tempo alla loro generalizzazione; il desiderio di rendere il modello il più generico possibile si è rivelato un lavoro finalizzato all'eliminazione dei particolari caratteristici di Mida Solutions, i quali comportavano un inutile appesantimento della descrizione.

### 4.2.3 Perchè un nuovo formalismo

Durante il lavoro di analisi, isolata la base del campo di applicazione, si è potuto migliorare ulteriormente l'accuratezza della ricerca del tool software arrivando a strumenti quali lo standard UML[1]. Esso impone i propri vincoli obbligando chi lo utilizza a rispettare regole a volte troppo restrittive. Per far fronte a queste limitazioni, si è pensato di impiegare quest'ultimo come base per un nuovo modello che si potesse adattare facilmente alle esigenze evidenziate. La ridefinizione o l'aggiunta di vincoli, anche se facilitano il lavoro, implica la perdita di notevoli vantaggi quali tool di sviluppo e mantenimento nel tempo, da parte di terzi, del modello. Si capisce quindi che la volontà di utilizzare modelli standardizzati ai quali imporre i limiti dell'analisi si è rivelata una soluzione non ottima.

La prima idea nata agli albori del progetto era informatizzare il sistema di gestione della qualità qualità impiegando un approccio web oriented, in modo tale da poter navigare tra i processi assimilando la loro struttura da file precedentemente creati dagli utenti finali, interagendo con pagine costruite ad hoc. Lo scheletro del processo sarebbe stato descritto semplicemente disegnandolo mediante un tool software basato sul nostro formalismo e successivamente esportato in un formato adatto ad una lettura automatica. Aggiungendo questo ulteriore paletto (tool capaci di esportare i disegni in un formato idoneo), la ricerca ha portato a dover scegliere tra l'uso di strumenti già esistenti, ma che generano file strutturali troppo difficili da gestire, oppure lo studio di un altro sistema per informatizzare il sistema di qualità.

Alla luce dei fatti, si è deciso di sacrificare i vantaggi dovuti all'impiego di uno standard già dotato di strumenti di gestione, a favore di un set di regole ed elementi, da noi definiti, che si potessero adattare perfettamente alla situazione presa in analisi. Questo modo di operare ha fatto nascere un formalismo estrema-

[1]UML (Unified Model Language) è un insieme di diagrammi formali e semiformali usato per aiutare il progettista informatico nella descrizione del problema e della soluzione. La prima versione di UML è stata sviluppata nel 1994 con lo scopo di fornire uno strumento unico di descrizione a supporto del lavoro dell'ingegnere del software. UML è continuamente aggiornato ed arricchito: la versione più recente è la 2.0.

mente flessibile e completo impiegabile mediante un qualsiasi software di image o process editing (nello specifico è stato impiegato Microsoft Visio). La scelta è stata accompagnata da una radicale modifica del sistema d'informatizzazione. Abbandonata l'idea di utilizzare file strutturati, si è continuato a rappresentare l'azienda come collezione di processi, costituiti da una sequenza di elementi base collegati tra di loro mediante relazioni di precedenza e relazioni tra dati; la nuova idea però prevede di adottare un sistema, quale una base di dati, per salvare tutti gli elementi di un processo e le relazioni che li legano tra di loro e con i dati, così che la navigazione di un processo si possa ridurre all'esecuzione di semplici query. L'impiego di un database comporta un ulteriore vantaggio: è, infatti, possibile instaurare una relazione diretta tra i dati strutturali e quelli runtime (informazioni che sono generate durante l'attraversamento di un processo). Si capisce, quindi, che la base di dati, che verrà illustrata nei capitoli successivi, è sostanzialmente divisa in due parti: una dedicata alla struttura dell'azienda e un'altra impiegata come punto di salvataggio di tutte le informazioni create durante l'utilizzo del sistema di gestione della qualità.

### 4.2.4 Descrizione del formalismo

La rappresentazione di un processo aziendale dev'essere realizzata mediante la stesura di un diagramma di control flow utilizzando gli elementi e le regole che seguono.

#### Start

Lo *Start* viene impiegato per identificare il punto di inizio del processo ed è fornito di più uscite. Nello stesso diagramma possono esistere più copie del blocco start così da facilitare la rappresentazione di processi molto complessi anche se esse vanno intese come un'unica entità. Sebbene vi possano essere più rappresentazioni di tale elemento l'abilitazione di uno di essi causerà l'attivazione di tutte le copie. Si può, quindi, dire che esso è logicamente unico.

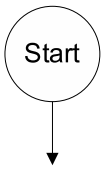


**Figura 4.1.** Rappresentazione grafica del blocco *Start*.

#### Attività

Il blocco *Attività* è un elemento fondamentale perché rappresenta l'azione che l'utente è chiamato a compiere per far si che l'attraversamento del diagramma

possa avanzare. Un'attività può produrre dati sensibili per il sistema, come, ad esempio, documenti, ed è, quindi, molto importante specificare quali dati sono ingresso e quali invece vengono prodotti. Si può inoltre intuire che una parte consistente di uno schema di processo aziendale è rappresentato da una "catena" di attività collegate tra loro da archi di precedenza, i quali, oltre a definire i percorsi, sottolineano il vincolo che impedisce di abilitare un'attività se quelle che la precedono non hanno terminato. Graficamente un'attività è rappresentata da un rettangolo dotato di:

- ingressi di flusso multipli;
- uscite di flusso multiple;
- dati in ingresso multipli;
- dati in uscita multipli.

La produzione di dati da parte di un'attività è un aspetto molto importante per l'evoluzione del processo e per questo essa non può essere considerata terminata se non ha prodotto tutti i dati in uscita; viceversa, un'attività non può essere abilitata se non dispone di tutti i dati in ingresso. Tali vincoli permettono di evitare possibili situazioni di stallo dovute alla mancanza di dati. Si crea, di conseguenza, oltre a un vincolo di precedenza, un vincolo sui dati.

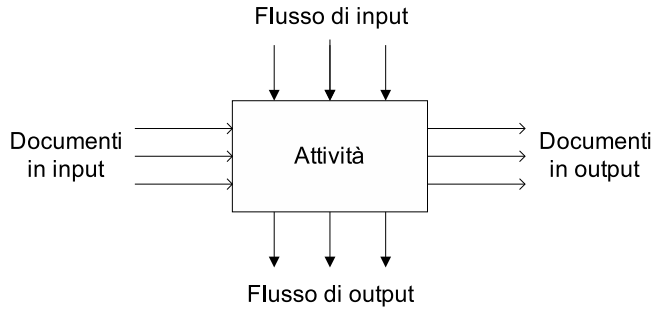


**Figura 4.2.** Rappresentazione grafica del blocco *Attività*.

### Scelta

Gli oggetti *Scelta* vengono impiegati per rappresentare un punto di decisione. Questo elemento permette di decidere il percorso di attraversamento del diagramma sulla base di una scelta; una volta che questa è stata presa, solo l'uscita di flusso ad essa associata sarà abilitata. Il blocco di scelta è rappresentato attraverso un rombo dotato di un unico ingresso di flusso e $n$ uscite, una per ogni possibile esito della decisione.

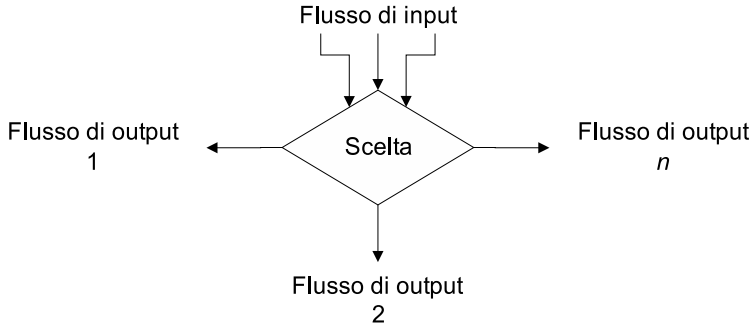


**Figura 4.3.** Rappresentazione grafica del blocco *Scelta*.

### Or

Il blocco *Or* permette di fondere due o più archi di precedenza per formarne uno unico. Tale blocco ha $n$ ingressi e un'unica uscita; il nome ne rappresenta la caratteristica base, esso, infatti, rispetta le regole dell'omonimo operatore logico, ovvero viene abilitata l'uscita quando almeno un ingresso è attivo.

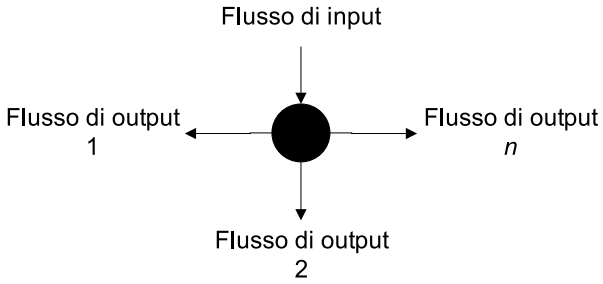


**Figura 4.4.** Rappresentazione grafica del blocco *Or*.

### Finish

Il *Finish* rappresenta il concetto di conclusione con successo che si manifesta solamente se il processo termina raggiungendo lo scopo per il quale è stato creato. Il blocco finish è dotato di $n$ ingressi e nessuna uscita e, come per il blocco *Start*, è possibile duplicarlo durante la stesura del diagramma per facilitare la schematizzazione dei processi complessi, ricordando, però, che esso è logicamente unico.

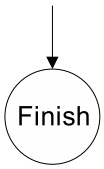


**Figura 4.5.** Rappresentazione grafica del blocco *Finish*.

### Stop

Diversamente dal finish, il blocco *stop* viene impiegato nel momento in cui il flusso di attraversamento incorre in un'eccezione che ne provoca la conclusione senza successo. Il blocco stop è dotato di $n$ ingressi e nessuna uscita e, come per il blocco *Start*, è possibile duplicarlo durante la stesura del diagramma per facilitare la schematizzazione dei processi complessi sempre ricordandone l'unicità logica.

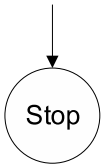


**Figura 4.6.** Rappresentazione grafica del blocco *Stop*.

### Subprocess

Questo elemento consente di passare ad un livello inferiore (livello attuale +1). Esistono, infatti, casi in cui un processo, per poter terminare, necessiti di dati che non gli competono dovendo, quindi, avviare un sottoprocesso che glieli possa fornire. Per questo motivo, la rappresentazione grafica di un processo aziendale può essere distribuita su più livelli. Il blocco è rappresentato da un rettangolo con due fasce laterali dotato di $n$ ingressi, logicamente in *and* tra di loro e un'unica uscita; all'interno del rettangolo sarà riportato il nome del sotto processo da avviare. Il passaggio ad un livello superiore (livello attuale -1) è implicitamente rappresentato dal blocco finish, infatti, a processo ultimato, per conoscere il percorso da seguire nella "catena di produzione" è necessario passare ad un livello meno dettagliato in cui sono descritte le relazioni tra processi (livello zero).

Oltre ai blocchi che compongono il flusso del processo, sono stati definiti anche degli oggetti fondamentali per l'interazione tra attività e utenti.

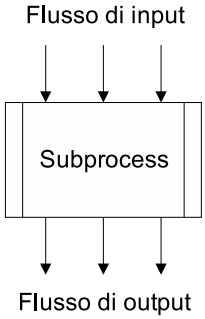


**Figura 4.7.** Rappresentazione grafica del blocco *Subprocess*.

### Documento

Rappresenta un elemento che ha sede in un repository oppure, più semplicemente, in un archivio interno all'azienda. Un documento può essere soggetto a revisioni ognuna identificata da un numero sequenziale.

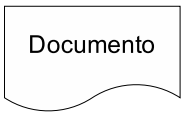


**Figura 4.8.** Rappresentazione grafica del blocco *Documento*.

### Registrazione

Rappresenta un elemento destinato a un supporto digitale, ad esempio un database, e per il quale non è prevista la revisione.


**Figura 4.9.** Rappresentazione grafica del blocco *Registrazione*.

### Notifica

È un documento che non può essere soggetto a revisione. Un esempio di notifica sono le e-mail inviate ai clienti per comunicare l'avvenuta apertura di un ticket.


**Figura 4.10.** Rappresentazione grafica del blocco *Notifica*.

### Documento multiplo

È un documento del quale non si conosce a priori il nome o la quantità di documenti che rappresenta. Esempio di documento multiplo sono i file sorgenti oppure la documentazione tecnica.


**Figura 4.11.** Rappresentazione grafica del blocco *Documento multiplo*.

Gli elementi descritti devono essere direttamente collegati alle attività utilizzando degli archi orientati grazie ai quali è possibile specificare se sono prodotti (Attività —> Documento/Registrazione/Notifica/Documento Multiplo) o utilizzati (Attività <— Documento/Registrazione/Notifica/Documento Multiplo). Nel diagramma di schematizzazione di un processo aziendale va riportato il responsabile di processo. In aggiunta a ciò vanno esplicitati i responsabili di ogni elemento del diagramma, cioè coloro che possono compiere l'azione riportata nelle attività o prendere decisioni nel caso di una scelta. La specifica del responsabile di elemento deve avvenire mediante delle bande funzionali, le quali riporteranno in una fascia laterale il ruolo del responsabile di elemento. La stesura del diagramma di control flow deve inoltre rispettare le seguenti scelte stilistiche:

- gli archi di precedenza devono avere uno spessore maggiore rispetto a quelli che collegano documenti, registrazioni e notifiche con le attività;
- il testo deve avere una dimensione che permetta una facile lettura senza quindi utilizzare font o dimensioni non adatte ad un documento tecnico;
- le bande funzionali devono essere disegnate impiegando una linea tratteggiata con spessore inferiore rispetto a quello scelto per gli archi di precedenza.

### 4.2.5 Esempi di applicazione

Per fare chiarezza sul formalismo appena descritto vengono mostrati alcuni esempi in vengono mappate, tramite le regole sopra definite, alcune sezioni dei processi aziendali che verranno illustrati in maniera approfondita nella prossima sezione.

In figura 4.12 si ha una sezione del processo *Product Management*, dove si osservano quattro tipi di oggetti differenti:

- subprocess;
- scelta;
- or;
- stop.

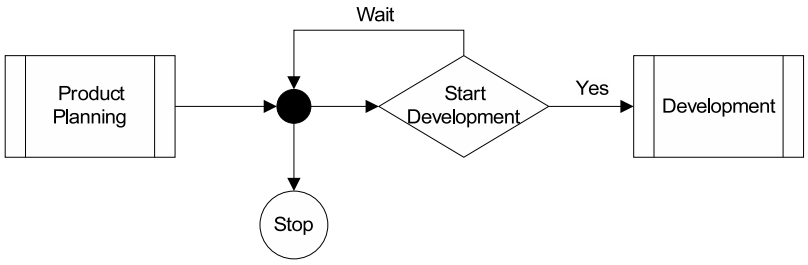


**Figura 4.12.** Esempio di utilizzo del formalismo tratto dal processo *Product Management*.

La sezione comincia con il sottoprocesso *Product Planning*: in questo caso il processo chiamante non avanza finchè il sottoprocesso non ha terminato la sua esecuzione. Successivamente l'Executive Board deve validare il progetto in via di sviluppo. Tale approvazione è rappresentata da un blocco di scelta con tre possibili uscite:

- se al momento non è possibile procedere allo sviluppo per qualche motivo il progetto viene messo in attesa;
- se non ci sono problemi si procede con il sottoprocesso *Development*;
- se l'Executive Board decide di non validare il progetto il processo termina prematuramente.

Analizzando in dettaglio le uscite dell'oggetto appena considerato si notano due particolari casi. Il primo è rappresentato dal ciclo nel diagramma di control flow: in questa circostanza, per ricongiungersi alla freccia di flusso entrante nel blocco *Scelta* viene utilizzato un oggetto di tipo *Or*. Di fatto, questo elemento di congiunzione è molto usato per creare iterazioni nell'esecuzione dei processi. Il

secondo caso particolare è rappresentato dalla terminazione del processo tramite blocco *Stop*.

In figura 4.13 si ha una sezione del processo Development, dove si notano cinque tipi di oggetti differenti:

- attività;
- scelta;
- documento;
- registrazione;
- or;

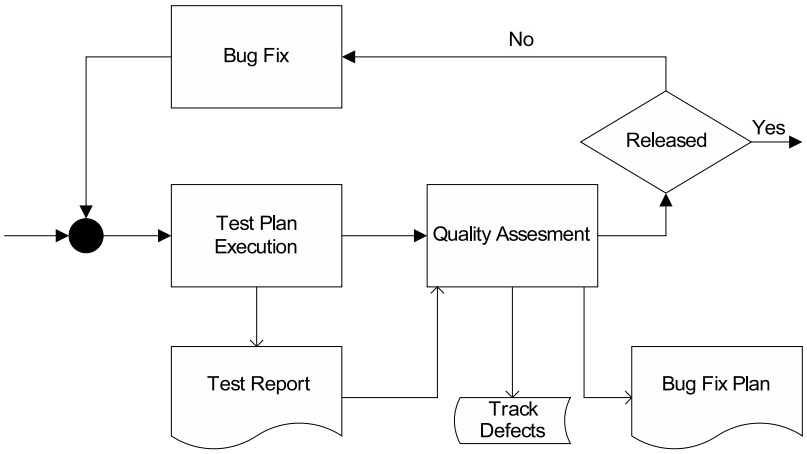


**Figura 4.13.** Esempio di utilizzo del formalismo tratto dal processo *Development*.

La sezione comincia con l'attività *Test Plan Execution*, che produce il documento *Test Report*. La porzione di processo continua con l'attività *Quality Assessment*, che riceve in ingresso il documento appena prodotto e restituisce in uscita *Bug Fix Plan* e *Track Defects*. Successivamente, si arriva ad un blocco di scelta dove si decide se rilasciare il prodotto o eseguire delle correzioni. In quest'ultimo caso viene avviata l'attività *Bug Fix* e la sezione ricomincia dall'inizio tramite un elemento di tipo *Or*.

### 4.2.6 Considerazioni

Durante la fase di ricerca del tool per il supporto al formalismo ed anche nella vera e propria ricerca di un modello da utilizzare sono state trovate molte soluzioni valide. Esse, però, si rivelavano, spesso, troppo ricche di informazioni rendendole, di conseguenza, difficili da apprendere. La volontà era quella di

realizzare un sistema che non crei disagi all'azienda che decide di utilizzarlo; sulla base di ciò, non si poteva sviluppare una soluzione che richiedesse uno step iniziale troppo impegnativo o che andasse a modificare la struttura aziendale. Per questi motivi, l'utilizzo di un modello generico, ma estremamente semplice (pochi elementi regolati da semplici vincoli), e di un sistema di salvataggio privo di regole (si possono creare tutte le relazioni desiderate una volta inseriti gli elementi da utilizzare) si rivela la soluzione più adatta. È importante ricordare che il lavoro di modellizzazione si è svolto dandogli il giusto tempo nella speranza di aver coperto e risolto tutti i punti critici che possono incorrere nella descrizione formale dei processi di un'azienda.

## 4.3 I processi aziendali

### 4.3.1 Sales Management

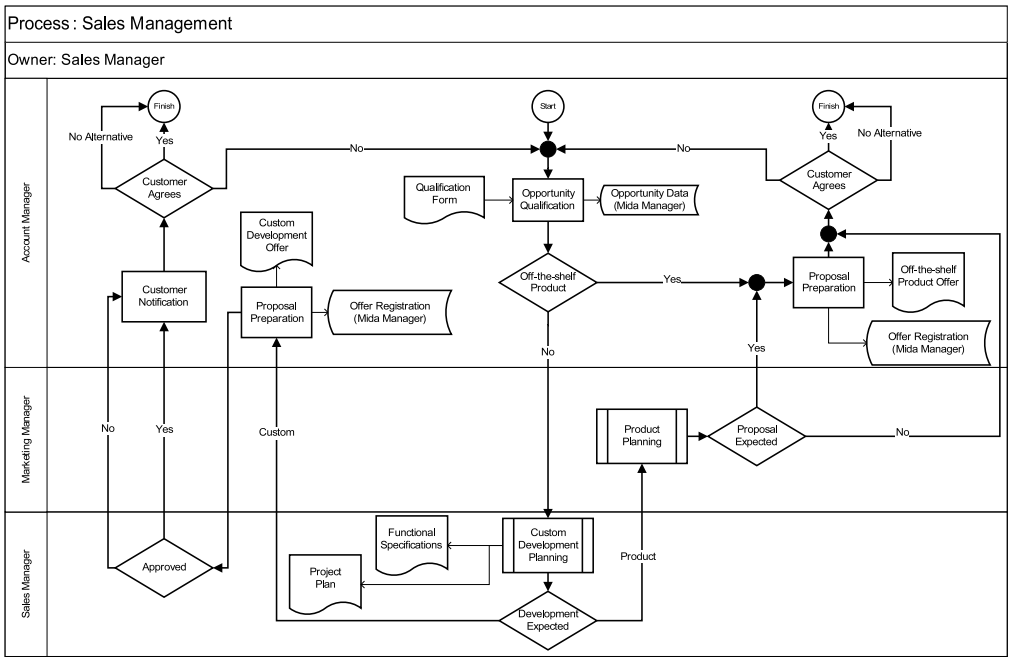


**Figura 4.14.** Rappresentazione grafica del processo Sales Management.

Il processo *Sales Management* (vedi figura 4.14) ha lo scopo di generare le offerte di prodotto sulla base delle richieste avanzate dai clienti. La fase di valutazione di quest'ultime può intraprendere percorsi diversi, in quanto le funzionalità richieste dal cliente potrebbero essere già offerte da un prodotto esistente, oppure semplicemente richiedere una personalizzazione se non come ultimo lo sviluppo di un nuovo prodotto. Al termine della prima fase, nella quale vengono

definite le funzionalità che il prodotto dovrà fornire, si può passare direttamente all'emissione dell'offerta se la richiesta del cliente ha permesso di identificare un prodotto già disponibile. In caso contrario, viene fatta una valutazione sul piano economico e tecnico, la quale prevede un'analisi di opportunità e di fattibilità, con cui viene deciso se realizzare un nuovo prodotto oppure apportare delle modifiche ad uno già esistente; in entrambi i casi sarà prodotta un'offerta per il cliente.

#### Documenti in ingresso e in uscita

In ingresso al processo si ha:

- *Qualification Form*: è il documento che viene compilato, sulla base di un template, nel momento in cui arriva la richiesta del cliente e che contiene tutte le informazioni utili per valutare le funzionalità del prodotto richiesto.

In uscita al processo si ha:

- *Off-the-shelf Product Offer*: è il documento che contiene l'offerta di prodotto, nel caso in cui il cliente abbia richiesto un prodotto già esistente;
- *Custom Development Offer*: è il documento che contiene l'offerta di prodotto, nel caso in cui venga realizzato un nuovo prodotto;
- *Functional Specifications*: le specifiche funzionali per lo sviluppo del prodotto; esso viene generato solamente se la richiesta del cliente non ha identificato un prodotto già esistente;
- *Project Plan*: è il documento che contiene il piano di progetto per lo sviluppo del prodotto; esso viene generato solamente se la richiesta del cliente non ha identificato un prodotto già esistente.

#### Particolarità del processo

La particolarità del processo si ha nella parte finale; si può notare che un'offerta "respinta" dal cliente non porta ad una modifica di quest'ultima, ma provoca la terminazione e la conseguente riattivazione del processo con in ingresso le nuove richieste. In questo modo ogni offerta di prodotto, anche se riferita alla stessa richiesta dello stesso cliente, non verrà mai perduta.

### 4.3.2 Custom Development Planning

Il processo *Custom Development Planning* (figura 4.15) ha la funzione di valutare le commesse, ovvero dei prodotti personalizzati su richiesta diretta dei clienti, sul piano economico e tecnico, prevede infatti un'analisi di opportunità e di fattibilità. Il processo nasce da una richiesta specifica di un cliente. A questo punto,

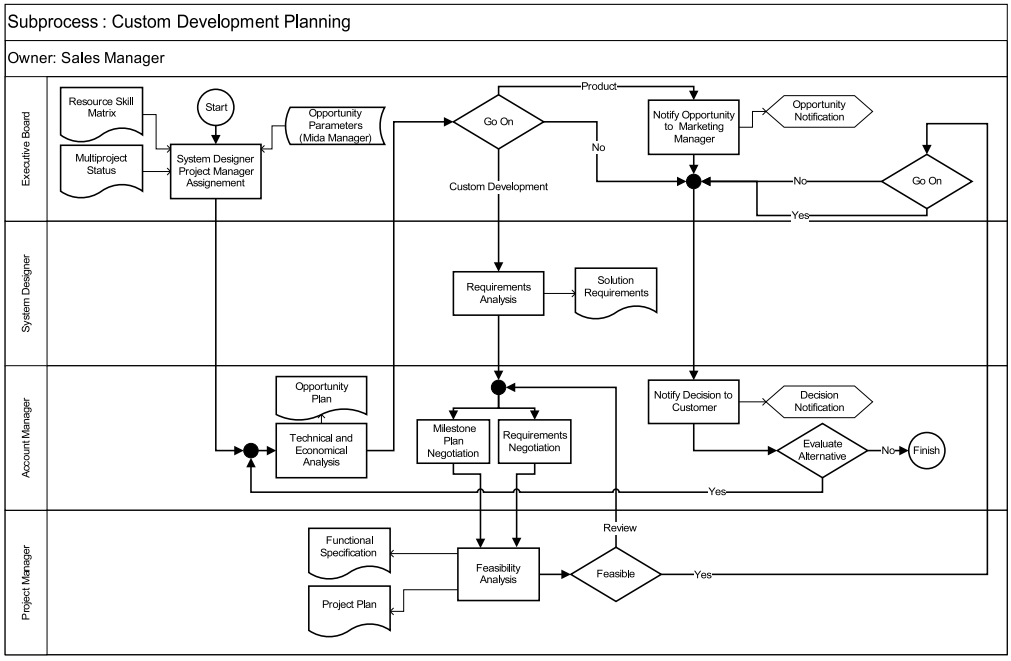


**Figura 4.15.** Rappresentazione grafica del processo Customer Development Planning.

viene assegnato un projet manager per gestire la commessa. Questo, preso contatto con il cliente, fa un'analisi tecnica-economica da cui nasce l'*Opportunity plan*. Mediante quest'ultimo documento la direzione decide se procedere oppure terminare il processo. In caso positivo, il project manager, conciliando le richieste del cliente con le sue possibilità in campo tecnico, produce un documento contenente i requisiti della propria soluzione. A questo punto, ripresi i contatti con il cliente, negozia i requisiti definitivi e i tempi di sviluppo. Superata l'analisi di fattibilità con le condizioni del cliente, la decisione finale spetta all'executive board : sia in caso che si scelga di procedere con lo sviluppo o meno, il processo termina con una notifica al cliente, il quale potrà, in caso di rifiuto, rivedere la sua richiesta.

### Documenti in ingresso e in uscita

I documenti in ingresso sono :

- *Resource Skill Matrix*: contiene informazioni a proposito delle capacità specifiche dei singoli sviluppatori

- *Multiproject Status*: contiene informazioni sullo stato dei progetti in corso d'opera

- *Opportunity Parameters*: contiene informazioni a proposito dei prodotti già presenti in listino

I documenti in uscita sono :

- *Opportunity Plan*: contiene un'analisi economica e tecnica della richiesta del cliente
- *Solution Requirements*: contiene i requisiti della *bozza* di prodotto, in quanto questi verranno ridiscussi con il cliente
- *Functional Specification*: contiene le specifiche funzionali per lo sviluppo del prodotto custom
- *Project Plan*: contiene il piano di progetto per lo sviluppo del prodotto custom

#### Particolarità del processo

A differenza dello sviluppo prodotti, queste soluzioni su misura per clienti con specifiche richieste sono portate a termine in genere da un solo sviluppatore che viene scelto in base alla sua disponibilità e alle sue capacità. Dopo l'approvazione dell'*opportunity plan*, può capitare che venga alla luce una buona idea per un prodotto : in questo caso il processo termina con una notifica di rifiuto al cliente che ne aveva effettuato la richiesta e un'altra al marketing che segnala l'opportunità di valutare un nuovo prodotto generico. La produzione dei requisiti del prodotto custom non si basa solo sulle richieste del cliente, ma su un compromesso tra queste e la fattibilità di esse sul piano tecnico. Dopo aver definito le caratteristiche della soluzione proposta dallo sviluppatore, queste vengono discusse e concordate con il cliente.

### 4.3.3 Product Planning

Il processo *Product planning* (figura 4.16) si occupa di effettuare un'analisi approfondita sull'opportunità di creazione di un nuovo prodotto. Se questa mette in evidenza buone possibilità di vendita e guadagno, al termine del processo vengono definiti i parametri per lo sviluppo. Il processo comincia con la redazione del *business plan*, il quale dà un'indicazione all'executive board se continuare, magari rivedendo alcuni dettagli, con l'assegnazione del project manager. Dopo la creazione della *road map*, vengono definiti i requisiti del prodotto, ovvero le funzioni che questo dovrà implementare. A questo punto il lavoro procede sul piano del marketing con la creazione del *marketing plan*, e sul piano tecnico con la redazione di *specifiche funzionali* e *piano di progetto*. L'executive board, basandosi su questi ultimi tre documenti, decide se cominciare lo sviluppo anche revisionando certi documenti e procedure, oppure se fermare il processo e non procedere alla realizzazione del prodotto.

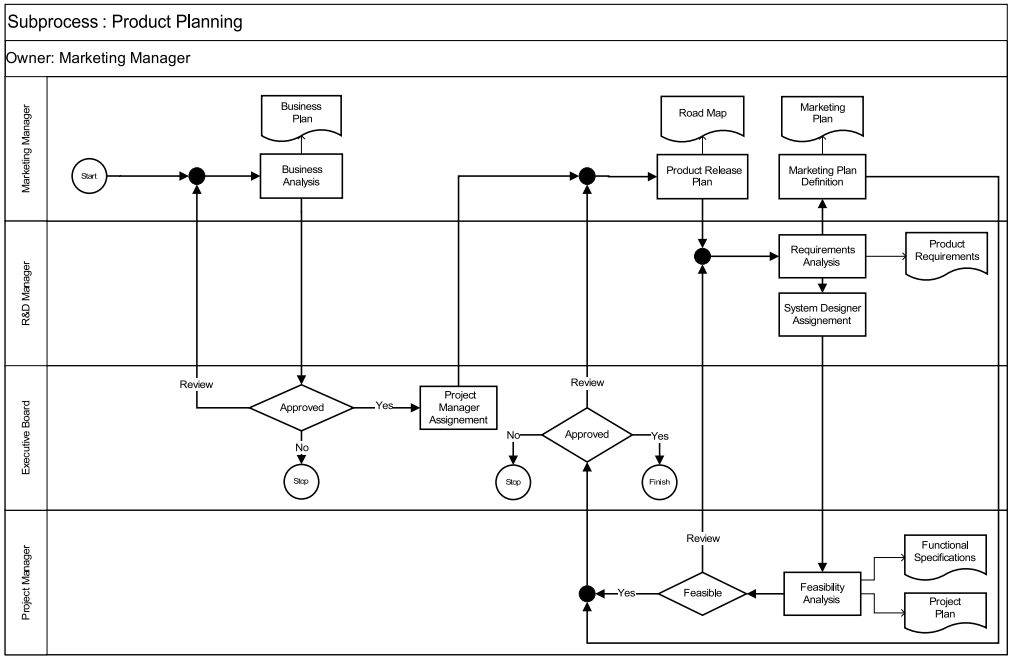


**Figura 4.16.** Rappresentazione grafica del processo Product Planning.

### Documenti in ingresso e in uscita

Per questo processo non sono previsti documenti in ingresso. In uscita ne troviamo diversi :

- *Business Plan*: contiene una previstione dei tempi di sviluppo, dei costi e dei possibili profitti
- *Road Map*: contiene un diagramma temporale che dà un'indicazione sui tempi di sviluppo del prodotto
- *Marketing Plan*: contiene i prezzi, il piano di comunicazione e la definizione dei canali di vendita
- *Product Requirements*: contiene i requisiti che il prodotto deve soddisfare
- *Functional Specification*: contiene le specifiche funzionali per lo sviluppo
- *Project Plan*: contiene il piano di progetto per lo sviluppo

### Particolarità del processo

La *Road map* è uno documento molto utile, perchè permette sia al marketing, che deve presentare il prodotto sul mercato, sia agli sviluppatori, che devono

lavorare su diverse parti del progetto, di avere delle scadenze temporali abbastanza precise. In realtà, durante l'avanzamento del progetto, questo documento viene revisionato molte volte, anche in base a cambiamenti in corso d'opera sul prodotto stesso. Un'altro aspetto da evidenziare è una delle fasi finali del processo : prima di sottoporre il progetto al giudizio finale dell'executive board, sono i tecnici stessi che ne fanno un'analisi di fattibilità e possono richiedere una revisione dei requisiti di prodotto, senza interpellare direttamente la direzione, nel caso il progetto da sviluppare non sia fattibile.

### 4.3.4 Product Management

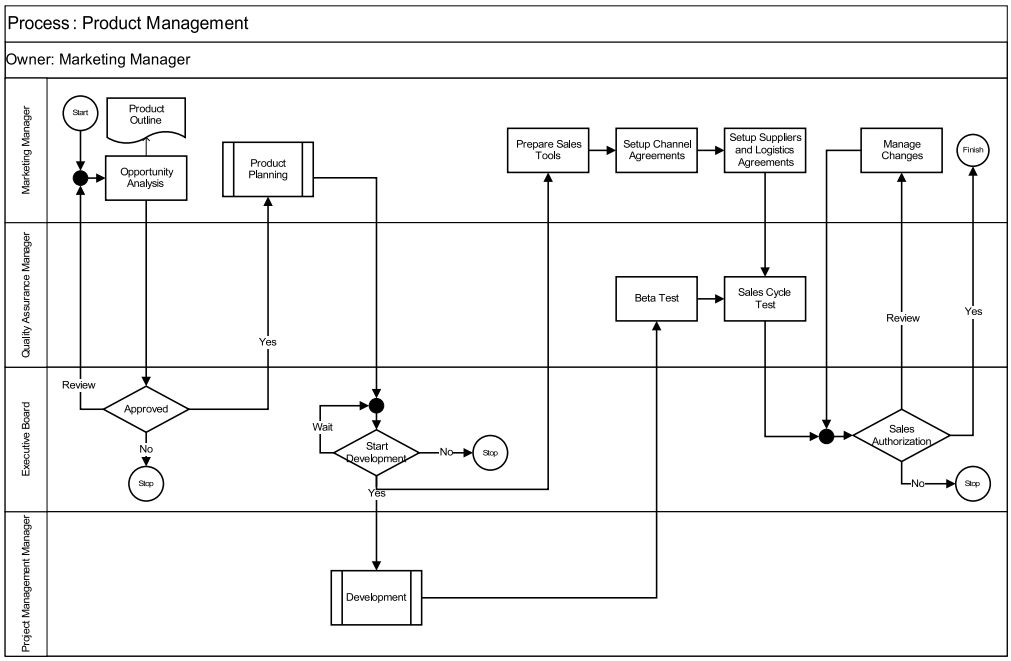


**Figura 4.17.** Rappresentazione grafica del processo Product Management.

Il processo *Product management* (figura 4.17) descrive l'intero ciclo di creazione di un prodotto, dall'analisi di opportunità al lancio sul mercato. Il processo comincia con un'analisi d'opportunità, ovvero una valutazione generale di un'*idea* per un nuovo prodotto. Se questa viene accettata viene eseguito il sottoprocesso *Product planning* che esamina più a fondo le caratteristiche che deve avere il nuovo prodotto e dà le direttive per il suo sviluppo. Dopodichè cominciano simultaneamente i beta test sul piano tecnico, mentre il marketing si occupa di accordarsi con i canali di vendita per il lancio del prodotto. Infine, dopo l'approvazione dell'executive board, il prodotto viene rilasciato nel mercato. In caso di revisione da parte della executive board al momento dell'autorizzazione,

il marketing manager deve effettuare i cambiamenti proposti e far rivalutare il prodotto prima di procedere ad un eventuale rilascio nei canali di distribuzione.

### 4.3.5 Development

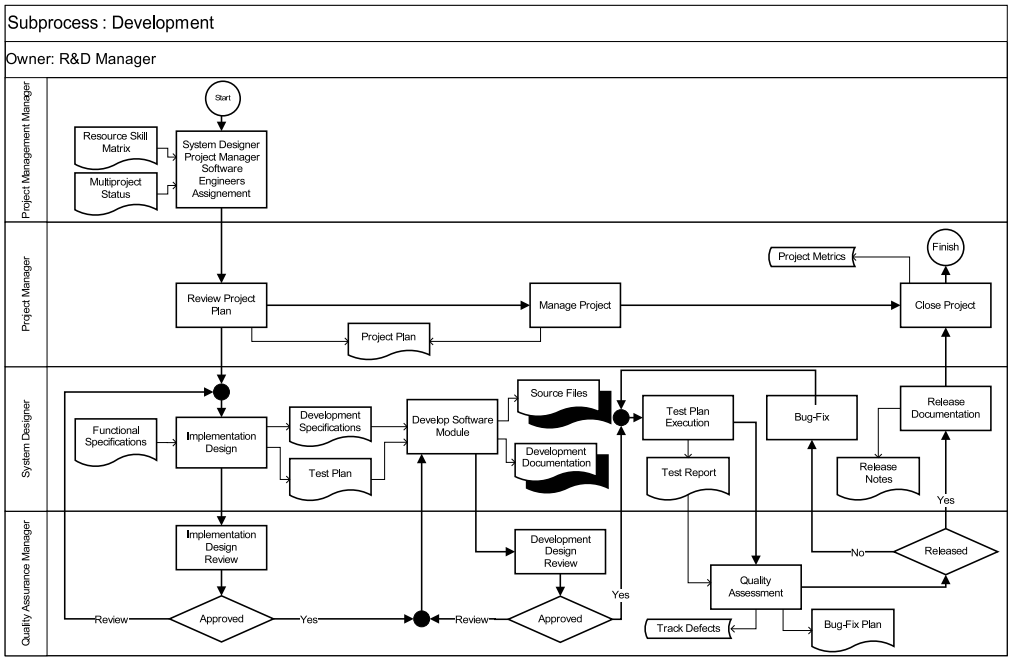


**Figura 4.18.** Rappresentazione grafica del processo Development.

Il processo *Development* (figura 4.18) ha come scopo lo sviluppo di un prodotto e quindi la realizzazione del software corredato dalla documentazione tecnica e di release. La fase iniziale del processo prevede un'analisi del personale, cioè una valutazione delle competenze e della disponibilità di ogni sviluppatore, così da poter selezionare il personale che costituirà la risorsa umana del progetto. A seguito di questa prima fase, vi è la definizione delle specifiche di sviluppo (costituisce la suddivisione, per priorità, delle funzione da sviluppare che verranno successivamente distribuite sull'asse temporale) e la pianificazione della fase di test. Terminata l'analisi, si passa allo sviluppo del prodotto durante il quale viene realizzata, oltre al codice sorgente, la documentazione tecnica. Conclusa quest'ultima fase, viene compiuto un lavoro di test sul materiale prodotto generando, a bug risolti, release del prodotto le quali saranno completate dalla documentazione di release prima del rilascio.

#### Documenti in ingresso e in uscita

In ingresso al processo si ha:

- *Resource Skill Matrix*: è il documento che contiene una matrice di competenze la quale riporta tutte le competenze di ogni individuo del personale;
- *Multiproject Status*: è il documento dal quale è possibile conoscere lo stato d'avanzamento di ogni progetto;
- *Functional Specifications*: è il documento che contiene tutte le informazioni utili sulle funzionalità che il prodotto dovrà fornire.

In uscita al processo si ha:

- *Project Plan*: è il documento che contiene il piano di progetto per lo sviluppo del prodotto;
- *Development Specifications*: è il documento che contiene il piano di sviluppo e quindi la suddivisione, per priorità, delle funzione da sviluppare e con relativa distribuzione sull'asse temporale;
- *Source Files*: costituiscono i file sorgenti del porgetto;
- *Development Documentation*: costituisce la documentazione tecnica del progetto;
- *Test Plan*: è il documento che contiene il piano di test e quindi la divisione temporale delle funzionalità da verificare;
- *Test Report*: è il documento che contiene il report di ogni test così da conservare lo storico del testing;
- *Bug-Fix Plan*: è il documento che contiene la suddivisione, per priorità, dei bug da risolvere e con relativa distribuzione sull'asse temporale;
- *Release Notes*: è il documento che contiene tutte le informazioni della release alle quali sono associate.

### 4.3.6 Customer Support

Il processo *Customer Support* (figura 4.19) ha lo scopo di dare assistenza sui prodotti una volta che questi vengono venduti al cliente; l'assistenza viene garantita solo in presenza di un contratto di assistenza valido. Quindi, tutte le segnalazioni di mal funzionamenti o richieste di assistenza rappresentano l'input di questo processo; da notare che quasi sempre l'azienda non tratta direttamente con il cliente ma con i canali di vendita intermedi. Una volta che la segnalazione è giunta, viene fatta una prima analisi per cercare di individuare il probema; fatto questo, viene aperto un ticket all'interno di un sistema dedicato in modo da tracciare il lavoro che verrà eseguito. Se il contratto di assistenza è presente e ancora valido, si cerca di risolvere il problema e una volta risolto se ne chiede una conferma da parte del cliente; se tutto è a posto si chiude il ticket aperto all'inzio del processo e questo termina.

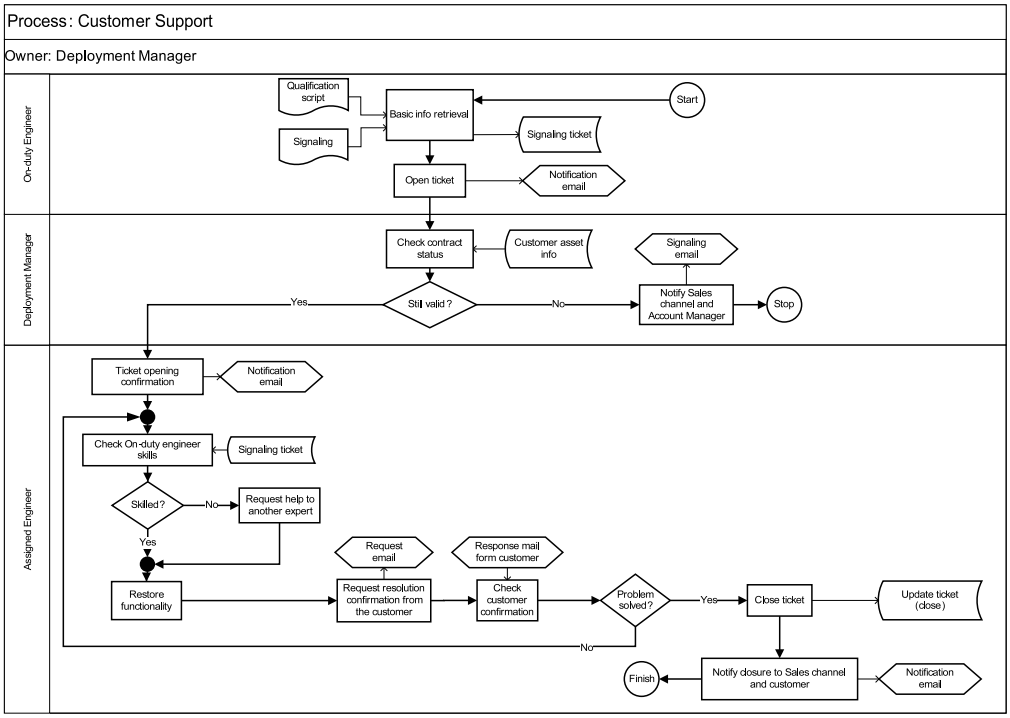


**Figura 4.19.** Rappresentazione grafica del processo Customer Support.

### Documenti in ingresso e in uscita

In ingresso al processo si hanno due documenti:

- *Signaling* : è il documento contenente la segnalazione da parte del cliente (o più spesso dal canale di vendita) di un guasto o comunque di un qualcosa che richiede l'intervento dei tecnici dell'azienda. La segnalazione generalmente arriva via mail oppure via fax, l'importante ovviamente è che sia una segnalazione scritta;

- *Qualification Script* : è un documento contenente una serie di domande standard che hanno l'obiettivo di ottenere quante più informazioni possano essere utili ad individuare e risolvere il problema riscontrato (alcune informazioni tipiche che si cercano di ottenere attraverso il questionario sono il riferimento del cliente, il tipo di problema riscontrato oppure che genere di informazioni si vuole).

In uscita non si ha nessun documento ma solo l'aggiornamento del ticket chiudendolo e la mail di notifica verso il cliente per notificargli l'avvenuta chiusura della sua segnalazione.

### Particolarità del processo

I passaggi importanti di questo processo si hanno all'inizio quando viene aperto il ticket della segnalazione; il ticket permetterà di tracciare la segnalazione e tutto il lavoro che ne deriverà; inoltre, l'identificativo del ticket viene comunicato anche al cliente. Altro punto molto importante, dato che determina se un cliente ha diritto ad avere assistenza oppure no, è il controllo di validità del contratto di assistenza: se questo è ancora valido allora si procede con la ricerca di una soluzione al problema, altrimenti verrà fatta una segnalazione al canale di vendita (che tratta con il cliente) e al commerciale interno all'azienda. Ultimo passaggio importante è quello alla fine del processo, una volta che è stato risolto il problema; infatti viene richiesta una conferma da parte del cliente della reale risoluzione del problema e in caso affermativo si comunica a quest'ultimo la chiusura del ticket relativo alla sua segnalazione.

## 4.3.7 Order Management

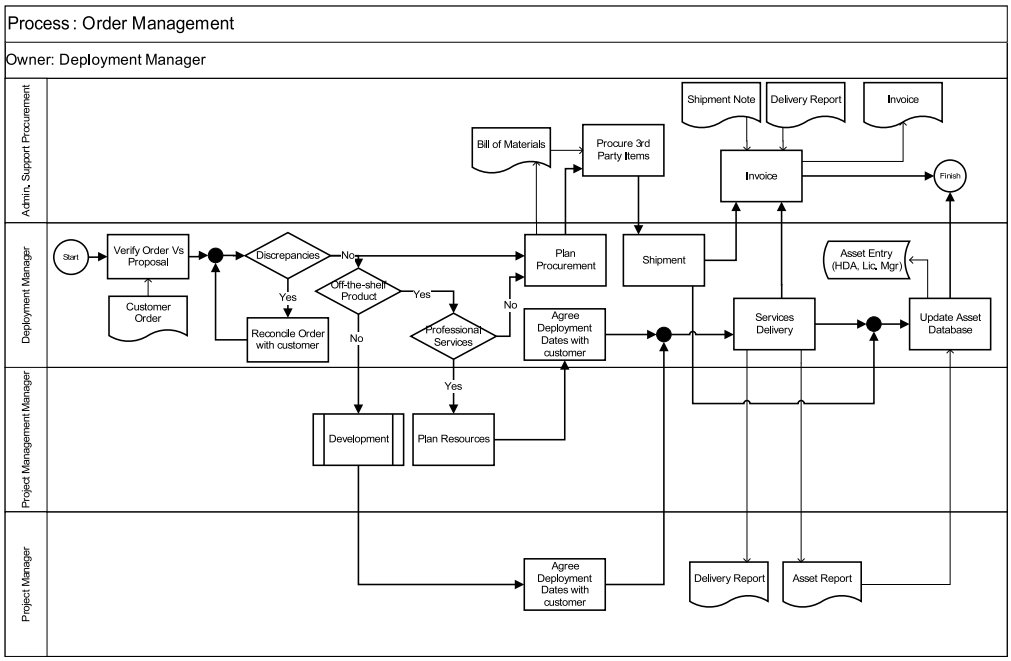


**Figura 4.20.** Rappresentazione grafica del processo Order Management.

Il processo *Order Management* (figura 4.20) ha lo scopo di predisporre la spedizione dei prodotti e organizzare gli eventuali servizi preofessionali verso i clienti che ne fanno richiesta; per servizi professionali si intendono ad esempio dei corsi tenuti da tecnici dell'azienda per insegnare a usare un particolare prodotto oppure la disponibiltà di tecnici per un certo periodo verso il cliente. Una

volta ricevuto l'ordine si fa un controllo sulla correttezza di questo ed eventuali differenze vengono discusse con il cliente; una volta che l'ordine è a posto si passa al reperimento del materiale se questo è disponibile oppure al suo sviluppo se così non fosse ed inoltre, se richiesti, si procede anche nella pianificazione del calendario per i servizi professionali (calendario che verrà poi discusso anche con il cliente). Dopo aver preparato tutti i documenti si passa alla spedizione e all'erogazione dell'eventuale servizio richiesto; infine, si avranno di ritorno la fattura e un resoconto su quello che è stato fatto durante il servizio.

### Documenti in ingresso e in uscita

In ingresso al processo troviamo i seguenti documenti:

- *Customer Order* : è il documento contenente la lista dei prodotti richiesti dal cliente; una volta verificata la sua correttezza si passa alla fase successiva, ovvero al reperimento del materiale vero e proprio.

- *Bill of Materials* : viene prodotto e poi usato all'interno del processo, per questo motivo si trova sia in ingresso che in uscita ad esso. Questo documento (derivato dal documento precedente ovvero *Customer order*) contiene la lista definitiva dei prodotti richiesti dal cliente; ci si basa su di esso per il reperimento del materiale vero e proprio;

- *Shipment Note* : documento contenente le note riguardanti la spedizione del materiale; esso è utile in fase di fatturazione del materiale;

- *Delivery Report* : viene prodotto e poi usato all'interno del processo, per questo motivo si trova sia in ingresso che in uscita ad esso. Questo documento è la ricevuta che attesta che il materiale è stato correttamente consegnato al cliente; inoltre, nel caso in cui sia stato anche richiesto un servizio professionale, in questo documento è anche presente un resoconto su che cosa è stato fatto.

In uscita al processo invece possiamo trovare più documenti:

- *Bill of materials* : viene prodotto e poi usato all'interno del processo, per questo motivo si trova sia in ingresso che in uscita ad esso. come spiegato prima, questo contiene la lista definitiva dei prodotti richiesti dal cliente; ci si basa su di esso per il reperimento del materiale vero e proprio.

- *Delivery report* : viene prodotto e poi usato all'interno del processo, per questo motivo si trova sia in ingresso che in uscita ad esso. Questo documento è la ricevuta che attesta che il materiale è stato correttamente consegnato al cliente; inoltre, nel caso in cui sia stato anche richiesto un servizio professionale, in questo documento è anche presente un resoconto su che cosa è stato fatto;

- *Invoice (Fattura)* : è un documento fiscale che viene emesso dall'azienda e che attesta che è stato venduto un qualcosa ad un particolare cliente e che quindi essa ha il diritto ad essere pagata per questo;

- *Asset report* : documento contenente la lista dei software che sono stati installati sulla macchina del cliente da parte del tecnico; questo documento è relativo solo al caso in cui ci sia stato un servizio professionale.

Oltre a questi documenti, prima della conclusione del processo, avviene l'aggiornamento o la creazione di un entry su database dedicati per tenere traccia del software installato sulla macchina del cliente nel caso in cui questo sia stato installato dai tecnici.

#### Particolarità del processo

Dopo che la lista degli acquisti è stata confermata può capitare che uno o più prodotti non siano disponibili o perchè sono esaurite le scorte oppure perchè sono prodotti ancora non esistenti; nel secondo caso si passa al processo di *Development* nel quale si andrà a progettare e poi, dopo tutte le valutazioni del caso, a produrre il prodotto richiesto.

### 4.3.8 Bug Management

L'obiettivo del processo *Bug Management* (figura 4.21) è gestire la soluzione dei bug software che vengono segnalati dall'assistenza clienti o dal reparto di sviluppo. Il bug da correggere viene sottoposto ad un'analisi di rischio (vedi figura 4.22) che si basa essenzialmente su due parametri: la probabilità che il bug si verifichi (intesa come frequenza) e l'impatto che il bug ha sul cliente. Come risultato dell'analisi, viene associato al bug un livello di rischio in base al quale si deciderà come operare. I livelli vengono determinati come segue:

- *Livello I*: probabilità e impatto bassi. I bug in questa categoria hanno un livello di rischio minimo e, quindi, richiedono un intervento a bassa priorità. Questi verranno, quindi, risolti solamente quando saranno concluse le attività di sviluppo a priorità maggiore.

- *Livello II*: alta probabilità ma basso impatto. Anche se l'impatto di questi bug sul cliente non è alto, l'alta frequenza con cui essi si presentano suggerisce un intervento non troppo lento. In questo caso viene, quindi, aggiornata la *Roadmap* stabilendo quando si andrà ad intervenire per risolvere il problema.

- *Livello III*: probabilità bassa con alto impatto. Come con il livello di rischio precedente, si procede con l'aggiornamento della *Roadmap*, ma, in questo caso, la volontà è quella di risolvere il bug entro la data di rilascio della release successiva del prodotto.

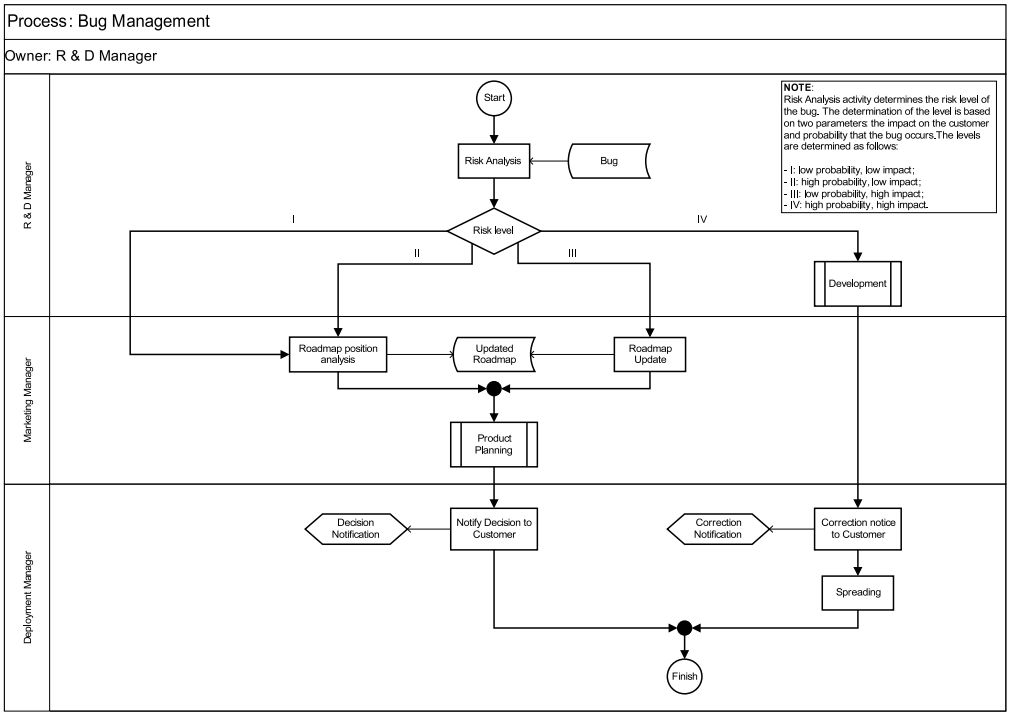


**Figura 4.21.** Rappresentazione grafica del processo Bug Management.

- *Livello IV*: probabilità e impatto alti. I bug che ricadono in questa categoria richiedono un intervento immediato da parte dell'azienda. Viene subito attivato un processo di sviluppo della patch correttiva che andrà, poi, distribuita ai clienti.

Ritornando alla procedura seguita per la risoluzione dei bug con i primi tre livelli di rischio, in seguito all'aggiornamento della *Roadmap*, viene attivato il processo di *Project Planning*, in seguito al quale viene notificata al cliente la decisione presa dall'azienda.

### 4.3.9 Supply Management

Il processo di *Supply Management* (figura 4.23) si occupa della gestione degli approvvigionamenti. In seguito all'evasione di un ordine può emergere la necessità di effettuare degli acquisti che porterà, quindi, alla stesura di una *Purchasing list*. Questa lista viene visionata dalla direzione che decide se approvarla o meno. In caso di approvazione viene effettuato un ordine al/ai fornitore/i. Nel caso un fornitore segnali dei possibili ritardi, viene avvisato il responsabile della commessa da evadere che si occuperà di effettuare le opportune azioni correttive (ad esempio contattare il cliente dell'azienda interessato per negoziare su una

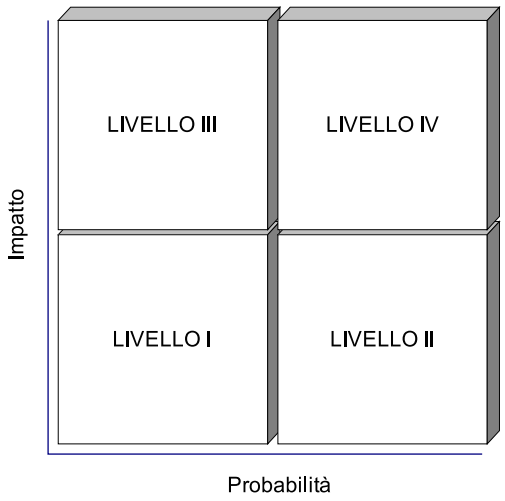


**Figura 4.22.** Schema per la determinazione del livello di rischio di un bug.

variazione delle scadenze). In caso contrario, invece, al ricevimento dei prodotti dei fornitori, si procede con il controllo dei documenti di trasporto e della fattura (documenti che verranno, poi, archiviati) ed, in seguito, con il controllo della merce. Se si riscontrano dei problemi in queste due attività, si procede contattando il fornitore, altrimenti si avvisa il responsabile della commessa dell'avvenuto approvvigionamento e si chiude il processo.

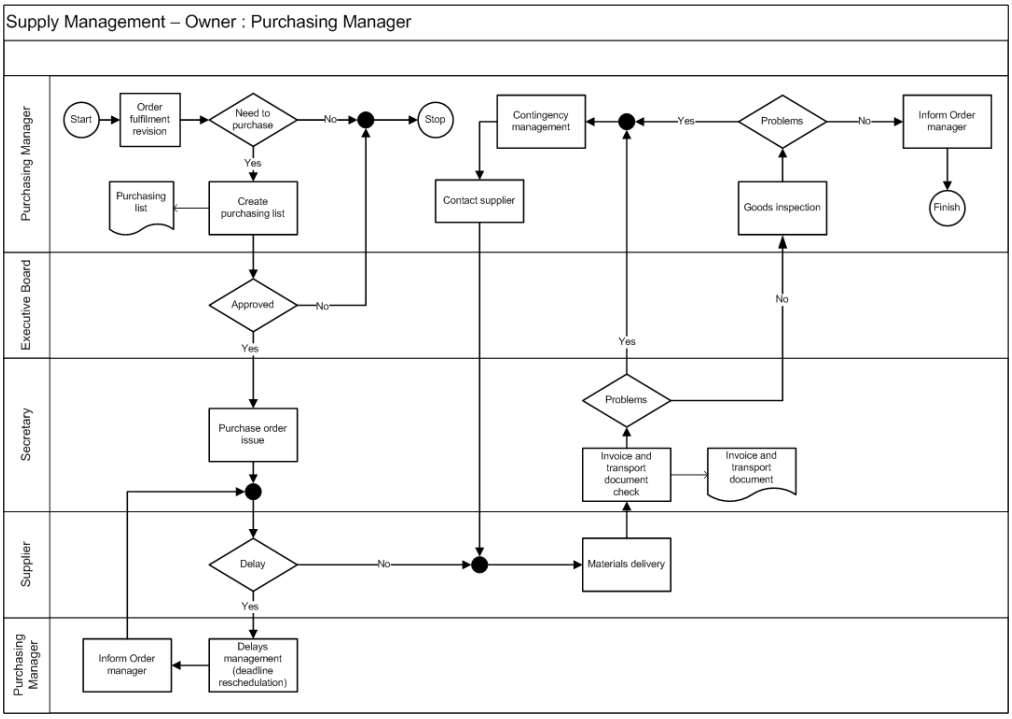


**Figura 4.23.** Rappresentazione grafica del processo Supply Management.

# Capitolo 5

# Valutazione

In questo capitolo vengono illustrate le caratteristiche del modello di riferimento utilizzato per valutare il processo software di Mida Solutions ed i risultati della valutazione stessa.

## 5.1 Il modello di riferimento

Il primo criterio per la selezione di un modello compatibile è che esso sia compatibile con il modello di riferimento. La compatibilità è essenziale al fine di fornire un livello di comparabilità tra i risultati di diverse valutazioni tramite la massimizzazione dell'affidabilità di differenti approcci ed ottenere un maggior grado di uniformità nella presentazione dei risultati.

Il modello di riferimento definisce un set di processi d'ingegnerizzazione del software universali che sono fondamentali per la buona ingegnerizzazione del software e che comprendono le attività di buona pratica. Qualsiasi modello, per essere compatibile con il modello di riferimento, deve contenere come minimo una parte di questo ambito. Il modello può essere un sottoinsieme del modello di riferimento. Può anche essere un estensione del modello di riferimento che comprende tutti i processi predefiniti ed alcuni addizionali al di fuori dell'ambito standard.

Al fine di rendere un modello compatibile con quello di riferimento, deve esserne indirizzato l'obiettivo come definito nel modello di riferimento e l'ottenimento degli attributi di processo che costituiscono la dimensione delle capacità. Inoltre, per soddisfare i requisiti della norma ISO/IEC TR 15504, deve essere documentato il set di indicatori di performance di processo e di capacità che permette un giudizio sulle capacità di processo basato su evidenza oggettiva.

È essenziale che il valutatore abbia accesso ai dettagli della mappatura tra gli elementi del modello e quelli di riferimento. Un valutatore dovrebbe confermare l'effettiva significatività della mappatura.

Il modello per una valutazione può essere scelto dal valutatore oppure può essere stipulato dallo sponsor della valutazione (nel qual caso questa procedura

deve essere documentata). In entrambi i casi esistono dei criteri che assicurano che la selezione sia appropriata. La maggior considerazione da fare nella selezione del modello, presupposto che ogni modello è compatibile con quello di riferimento, è l'idoneità con il contesto della valutazione. I fattori principali nella selezione di un modello sono:

- l'ambito pianificato della valutazione;
- il settore industriale dell'organizzazione;
- il dominio di applicazione dei componenti software su cui si concentra la valutazione;
- l'integrazione di un percorso di miglioramento per aumentare la maturità dei processi;
- requisiti specifici per una forte comparabilità con altre valutazioni o organizzazioni.

Se esistono dei modelli sviluppati specificatamente per l'utilizzo in particolari settori industriali o per particolari domini di applicazioni, dovrebbero essere studiati. Se un'organizzazione vuole eseguire una valutazione in un'area non rappresentativa dei propri normali domini, si dovrebbe porre ulteriore attenzione sull'adeguatezza del modello.

Il modello fornisce le definizioni di base di processi e attributi che sono punti di riferimento per il giudizio delle performance di processo. Da questo segue che il valutatore competente dovrebbe conoscere profondamente il modello specifico utilizzato. Inoltre, visto che il modello contiene un esaustivo set di indicatori per le performance e le capacità di processo, esso è un punto di riferimento per il valutatore al fine di soddisfare i requisiti nella documentazione degli indicatori per la giustificazione dei giudizi.

### 5.1.1 Descrizione del modello

Nella Parte 5 della norma ISO/IEC TR 15504 é descritto un esempio di modello di valutazione che ne soddisfa i requisiti e supporta le performance di una valutazione tramite la fornitura di indicatori per la guida all'interpretazione degli obiettivi e attributi di processo definiti nella Parte 2 della normativa.

Il modello di riferimento raggruppa i processi in tre cicli di vita che contengono cinque categorie in accordo con il tipo di attività.
I **processi del ciclo di vita primario** consistono nelle categorie **Customer-Supplier** e **Engineering**:

- La categoria di processi **Customer-Supplier** è formata da processi che impattano direttamente il cliente, lo sviluppo del supporto e la traduzione del software per il cliente, inoltre si impegnano per il corretto utilizzo del software.

- La categoria di processi **Engineering** consiste in processi che specificano, implementano o mantengono direttamente il prodotto software, le sue relazioni con il sistema e la documentazione utente.

I **processi del ciclo di vita di supporto** comprendono la categoria **Support**:

- La categoria di processi **Support** consiste in processi che potrebbero essere utilizzati da altri processi (inclusi altri processi di supporto) a vari punti del ciclo di vita del supporto.

I **processi del ciclo di vita organizzativo** comprendono le categorie di processi di **Management** e **Organization**:

- La categoria di processi di **Management** consiste in processi che contengono pratiche di natura generica che potrebbero essere utilizzate da qualsiasi persona che gestisce qualsiasi tipo di progetto o processo all'interno del ciclo di vita del software.

- La categoria di processi **Organization** consiste in processi che stabiliscono gli obiettivi di business dell'organizzazione e processi di sviluppo, prodotti e gestione delle risorse che, quando utilizzati all'interno di progetti, aiuteranno l'organizzazione a raggiungere gli obiettivi di business.

| Categoria di processo | | Processo | | |
|---|---|---|---|---|
| ID | Titolo | ID | Titolo e (Tipo di processo) | |
| Processi del ciclo di vita primario | | | | |
| CUS | Categoria Customer-Supplier | | | |
| | | CUS.1 | | Acquisizione (base) |
| | | | CUS.1.1 | Preparazione acquisizione (componente) |
| | | | CUS.1.2 | Selezione fornitori (componente) |
| | | | CUS.1.3 | Monitoraggio fornitori (componente) |
| | | | CUS.1.4 | Accettazione clienti (componente) |
| | | CUS.2 | | Erogazione (base) |
| | | CUS.3 | | Ottenimento dei requisiti (nuovo) |
| | | CUS.4 | | Operazione (extended) |
| | | | CUS 4.1 | Utilizzo operazionale (componente esteso) |
| | | | CUS 4.2 | Supporto cliente (componente esteso) |
| ENG | Categoria Engineering | | | |
| | | ENG.1 | | Sviluppo (base) |
| | | | ENG.1.1 | Analisi e progettazione dei requisiti di sistema (componente) |
| | | | ENG.1.2 | Analisi dei requisiti del software (componente) |
| | | | ENG.1.3 | Progettazione del software (componente) |
| | | | ENG.1.4 | Costruzione del software (componente) |
| | | | ENG.1.5 | Integrazione del software (componente) |
| | | | ENG.1.6 | Test del software (componente) |
| | | | ENG.1.7 | Integrazione e testing del sistema (componente) |
| | | ENG.2 | | Manutenzione del sistema e del software (base) |

**Figura 5.1.** I processi e le categorie di processi (Ciclo di vita primario).

| Categoria di processo | | Processo | | |
|---|---|---|---|---|
| ID | Titolo | ID | Titolo e (Tipo di processo) | |
| Processi del ciclo di vita di supporto | | | | |
| SUP | Categoria Support | | | |
| | | SUP.1 | | Documentazione (esteso) |
| | | SUP.2 | | Gestione della configurazione (base) |
| | | SUP.3 | | Quality assurance (base) |
| | | SUP.4 | | Verifica (base) |
| | | SUP.5 | | Validazione (base) |
| | | SUP.6 | | Revisione congiunta (base) |
| | | SUP.7 | | Audit (base) |
| | | SUP.8 | | Gestione dei rischi (base) |
| Processi del ciclo di via organizzativo | | | | |
| MAN | Categoria Management | | | |
| | | MAN.1 | | Management (basic) |
| | | MAN.2 | | Gestione progetto (nuovo) |
| | | MAN.3 | | Gestione qualità (nuovo) |
| | | MAN.4 | | Gestione rischio (nuovo) |
| ORG | Categoria Organization | | | |
| | | ORG.1 | | Allinamento organizzativo (nuovo) |
| | | ORG.2 | | Processo di miglioramento (base) |
| | | | ORG.2.1 | Creazione del processo (componente) |
| | | | ORG.2.2 | Valutazione del processo (componente) |
| | | | ORG.2.3 | Miglioramento del processo (componente) |
| | | ORG.3 | | Gestione risorse umane (esteso) |
| | | ORG.4 | | Infrastrutture (base) |
| | | ORG.5 | | Misurazioni (nuovo) |
| | | ORG.6 | | Riuso (nuovo) |

**Figura 5.2.** I processi e le categorie di processi (Ciclo di vita di supporto e organizzativo).

La descrizione di ogni categoria di processo include la caraterizzazione dei processi che contiene seguita da una lista dei nomi dei processi.
I processi sono descritti in termini di sei componenti come definito in ISO/IEC TR 15504-2: Identificatore di processo, Nome, Tipo, Obiettivo, Output, Note.
Oltre a queste informazioni la dimensione di processo del modello di valutazione fornisce informazioni su:

- il set di pratiche base per i processi, fornendo una definizione delle attività necessarie a completare gli obiettivi di processo;
- il numero di prodotti in input e output associati ad ogni processo;
- le caratteristiche associate con ogni prodotto lavorativo.

Si ricorda che ci sono cinque diversi tipi di processo. Tre di primo livello (base, esteso e nuovo) e due di secondo livello (componente e componente esteso).

1. *Processo Base* identico negli intenti con i processi contenuti nel ISO/IEC 12207[1];
2. *Processo Esteso* espansione di processi contenuti nel ISO/IEC 12207;
3. *Processo Nuovo* al di fuori dell'ambito della norma ISO/IEC 12207;
4. *Processo Componente* gruppo di una o più attività della normativa ISO/IEC 12207 appartenenti allo stesso processo;
5. *Processo Componente Esteso* come sopra ma con materiale aggiuntivo.

## 5.2 Identificazione degli elementi del modello di valutazione

Nella tabella 5.1, sono riportati i risultati dell'attività di identificazione degli elementi del modello di valutazione nei processi di Mida Solutions.

Al fine di identificare le pratiche senza ambiguità e relazionarle con l'architettura del modello viene definita una nomenclatura per esse. La nomenclatura per le pratiche base facilita l'identificazione delle categorie di processi, dei processi che appartengono a ogni categoria, e delle pratiche base che appartengono ad ogni processo. Per le pratiche di management la nomenclatura facilita l'identificazione dei livelli di capacità, degli attributi di processo che appartengono ad ogni livello, e delle pratiche di management che appartengono ad ogni attributo di processo.

Ogni pratica è legata alla propria entità genitore, processo o attributo, attraverso uno schema di numerazione basato sugli identificatori di processi e attributi del modello di riferimento.

Ad ogni pratica è assegnato un identificatore che consiste in un codice alfanumerico composto da parti multiple. Per una pratica base, l'identificatore è nella forma: PC.PR.***BP***PN. I codici sono:

- PC identificatore di categoria di processo
- PR numero di processo (all'interno della categoria di processo)
- BP pratica base
- PN numero di pratica (all'interno del processo o attributo)

Per limitare, invece, la quantità di testo nella tabella, sono state associate delle sigle ai processi aziendale:

[1]ISO/IEC 12207:1995 Information technology - Software life cycle processes: Definisce un sistema per i processi del ciclo di vita del software assieme ad una terminologia ben definita. Contiene processi, attività e task che devono essere applicate durante l'acquisizione di un sistema che contiene software, di un prodotto software unico o di servizi software.

- SAL-MGMT Sales Management
- CUS-DEV-PL Customer Development Planning
- PROD-PL Product Planning
- PROD-MGMT Product Management
- DEV Development
- CUS-SUP Customer Support
- ORD-MGMT Order Management
- BUG-MGMT Bug Management
- SUP-MGMT Supply Management

La dicitura (*) indica un'attività che non è identificabile direttamente in un processo ma di cui si è sicuri faccia parte del sistema produttivo aziendale, in quanto emersa dalle interviste.

La dicitura (**) indica, invece, che si suppone si possa associare l'attività al processo indicato anche se non descritta esplicitamente.

Tabella 5.1: Mappatura del modello sui processi di Mida Solutions.

| **Processo base** | **Processo componente** | **Codice** | **Denominazione** | **Processo Mida Solutions** |
|---|---|---|---|---|
| *Customer Supplier* | | | | |
| CUS.1 Proc. acquisizione | Preparazione acquisizione | CUS.1.1.BP1 | Identificazione necessità | |
| | | CUS.1.1.BP2 | Definizione requisiti | |
| | | CUS.1.1.BP3 | Preparazione strategia di acquisizione | |
| | | CUS.1.1.BP4 | Definizione criteri di accettazione | |
| | Selezione del fornitore | CUS.1.2.BP1 | Definizione requisiti di acquisizione | |
| | | CUS.1.2.BP2 | Selezione fornitore | |
| | | CUS.1.2.BP3 | Preparazione e negoziazione contratto | |
| | Controllo fornitore | CUS.1.3.BP1 | Fornitura feedback al fornitore | |
| | | CUS.1.3.BP2 | Revisione sviluppo con fornitore | |
| | | CUS.1.3.BP3 | Controllo acquisizioni | |
| | | CUS.1.3.BP4 | Controllo fornitore | |
| | Accettazione cliente | CUS.1.4.BP1 | Valutazione prodotti consegnati | SUP-MGMT |
| | | CUS.1.4.BP2 | Accettazione prodotti consegnati | SUP-MGMT |
| CUS.2 Processo di fornitura | | CUS.2.BP1 | Preparazione risposta | ORD-MGMT |
| | | CUS.2.BP2 | Negoziazione contratto | ORD-MGMT |
| | | CUS.2.BP3 | Sviluppo del sistema o del software | |
| | | CUS.2.BP4 | Identificazione attributi per consegna e installazione di successo | ORD-MGMT |
| | | CUS.2.BP5 | Consegna e installazione del software | ORD-MGMT |
| CUS.3 Processo di ottenimento dei requisiti | | CUS.3.BP1 | Ottenimento requisiti e richieste dei clienti | SAL-MGMT |
| | | CUS.3.BP2 | Accordo sui requisiti | CUS-DEV-PL |
| | | CUS.3.BP3 | Istituzione linee guida su requisiti clienti | CUS-DEV-PL |
| | | CUS.3.BP4 | Gestione cambiamenti nei requisiti dei clienti | CUS-DEV-PL |
| | | CUS.3.BP5 | Comprensione delle aspettative dei clienti | CUS-DEV-PL |
| | | CUS.3.BP6 | Istituzione meccanismo di interrogazione da parte dei clienti | |
| CUS.4 Processo funzionale | Utilizzo funzionale | CUS.4.1.BP1 | Identificazione rischi funzionali | DEV |
| | | CUS.4.1.BP2 | Esecuzione di test su funzionalità | DEV |
| | | CUS.4.1.BP3 | Funzionamento del software | PROD-MGMT |
| | | CUS.4.1.BP4 | Revisione problemi funzionali del software | |
| | | CUS.4.1.BP5 | Soluzione problemi funzionali | DEV |
| | | CUS.4.1.BP6 | Gestione richieste degli utenti | BUG-MGMT |
| | | CUS.4.1.BP7 | Documentazione degli espedienti temporanei | BUG-MGMT |
| | | CUS.4.1.BP8 | Controllo capacità e servizio del sistema | |
| | Supporto cliente | CUS.4.2.BP1 | Fornitura formazione utenti | ORD-MGMT |
| | | CUS.4.2.BP2 | Instaurazione supporto prodotto | CUS-SUPP |
| | | CUS.4.2.BP3 | Controllo performance | |
| | | CUS.4.2.BP4 | Determinazione livello soddisfazione cliente | |
| | | CUS.4.2.BP5 | Comparazione con i competitori | |
| | | CUS.4.2.BP6 | Comunicazione della soddisfazione dei clienti | |

Continua alla pagina successiva

**Tabella 5.1 – continua da pagina precedente**

| Processo base | Processo componente | Codice | Denominazione | Processo Mida Solutions |
|---|---|---|---|---|
| *Engineering* | | | | |
| ENG.1 Proc. di sviluppo | Analisi e progettazione requisiti di sistema | ENG.1.1.BP1 | Identificazione requisiti di sistema | PROD-PL/CUS-DEV-PL |
| | | ENG.1.1.BP2 | Analisi requisiti di sistema | PROD-PL/CUS-DEV-PL |
| | | ENG.1.1.BP3 | Descrizione architettura di sistema | PROD-PL/CUS-DEV-PL |
| | | ENG.1.1.BP4 | Allocazione requisiti | PROD-PL/CUS-DEV-PL |
| | | ENG.1.1.BP5 | Sviluppo strategia delle distribuzioni | PROD-PL/CUS-DEV-PL |
| | | ENG.1.1.BP6 | Comunicazione requisiti di sistema | PROD-PL/CUS-DEV-PL |
| | | ENG.1.1.BP7 | Instaurazione tracciabilità | PROD-PL/CUS-DEV-PL |
| | Analisi requisiti software | ENG.1.2.BP1 | Specifica requisiti software | PROD-PL/CUS-DEV-PL |
| | | ENG.1.2.BP2 | Determinazione impatto su ambiente lavorativo | PROD-PL/CUS-DEV-PL |
| | | ENG.1.2.BP3 | Valutazione e validazione requisiti con cliente | CUS-DEV-PL |
| | | ENG.1.2.BP4 | Sviluppo criteri di validazione software | DEV |
| | | ENG.1.2.BP5 | Sviluppo strategia delle distribuzioni | PROD-PL |
| | | ENG.1.2.BP6 | Aggiornamento requisiti | DEV,CUS-DEV-PL |
| | | ENG.1.2.BP7 | Comunicazione requisiti software | DEV |
| | | ENG.1.2.BP8 | Valutazione requisiti software | DEV |
| | Progettazione del software | ENG.1.3.BP1 | Sviluppo progetto architettura del software | DEV |
| | | ENG.1.3.BP2 | Progettazione interfacce | DEV |
| | | ENG.1.3.BP3 | Verifica progetto software | DEV |
| | | ENG.1.3.BP4 | Sviluppo progetto dettagliato | DEV |
| | | ENG.1.3.BP5 | Instaurazione tracciabilità | DEV |
| | Costruzione del software | ENG.1.4.BP1 | Sviluppo unità software | DEV |
| | | ENG.1.4.BP2 | Sviluppo procedure di verifica unità | DEV |
| | | ENG.1.4.BP3 | Verifica unità software | DEV |
| | | ENG.1.4.BP4 | Instaurare la tracciabilità | DEV |
| | Integrazione del software | ENG.1.5.BP1 | Sviluppo strategia di integrazione del sw | DEV |
| | | ENG.1.5.BP2 | Sviluppo strategia di test di regressione degli oggetti software integrati | DEV |
| | | ENG.1.5.BP3 | Sviluppo test oggetti software integrati | DEV |
| | | ENG.1.5.BP4 | Test oggetti software integrati | DEV |
| | | ENG.1.5.BP5 | Integrazione degli oggetti software | DEV |
| | | ENG.1.5.BP6 | Test regressione negli oggetti sw integrati | DEV |
| | Test del software | ENG.1.6.BP1 | Sviluppo strategia di test del software integrato, inclusa strategia di regressione | DEV |
| | | ENG.1.6.BP2 | Sviluppo dei test per software integrato | DEV |
| | | ENG.1.6.BP3 | Test di software integrato | DEV |
| | | ENG.1.6.BP4 | Test di regressione per software integrato | DEV |
| | Integrazione del sistena e di test | ENG.1.7.BP1 | Sviluppo strategie di integrazione del sistema e di test | DEV |
| | | ENG.1.7.BP2 | Sviluppo strategia dei test di regressione | DEV |
| | | ENG.1.7.BP3 | Costruzione aggregati delle unità di sistema | DEV |
| | | ENG.1.7.BP4 | Sviluppo test per le aggregazioni di sistema | DEV |
| | | ENG.1.7.BP5 | Test degli aggregati di sistema | DEV |
| | | ENG.1.7.BP6 | Sviluppo dei test per il sistema | DEV |
| | | ENG.1.7.BP7 | Test del sistema integrato | DEV |
| | | ENG.1.7.BP8 | Test di regressione sugli aggregati o sul sistema integrato | DEV |
| ENG.2 Processo di manutenzione del software e del sistema | | ENG.2.BP1 | Determinazione requisiti di manutenzione | |
| | | ENG.2.BP2 | Sviluppo strategia di manutenzione | |
| | | ENG.2.BP3 | Analisi problemi utenti e miglioramenti | CUS-DEV-PL |
| | | ENG.2.BP4 | Determinazione modifiche per prossimo upgrade | BUG-MGMT |
| | | ENG.2.BP5 | Implementazione e test modifiche | DEV |
| | | ENG.2.BP6 | Upgrade sistema utente | BUG-MGMT |
| | | ENG.2.BP7 | Ritiro del sistema utente | CUS-SUP(**) |
| *Support* | | | | |
| SUP.1 Processo di documentazione | | SUP.1.BP1 | Sviluppo politica di documentazione | |
| | | SUP.1.BP2 | Instaurazione standard per documentazione | |
| | | SUP.1.BP3 | Specifica dei requisiti di documentazione | |
| | | SUP.1.BP4 | Sviluppo documentazione | |
| | | SUP.1.BP5 | Verifica documentazione | |
| | | SUP.1.BP6 | Distribuzione documentazione | |
| | | SUP.1.BP7 | Mantenimento documentazione | |
| SUP.2 Processo di configurazione del management | | SUP.2.BP1 | Sviluppo strategia di configurazione del management | (*) |
| | | SUP.2.BP2 | Instaurazione sistema di configurazione del management | (*) |
| | | SUP.2.BP3 | Identificazione oggetti di configurazione | (*) |
| | | SUP.2.BP4 | Mantenimento descrizione oggetti di configurazione | (*) |
| | | SUP.2.BP5 | Gestione cambiamenti | (*) |
| | | SUP.2.BP6 | Gestione distribuzioni prodotto | (*) |
| | | SUP.2.BP7 | Mantenimento storico oggetti di configurazione | (*) |
| | | SUP.2.BP8 | Documentazione stato della configurazione | (*) |
| | | SUP.2.BP9 | Gestione distribuzioni e delle consegne degli oggetti di configurazione | (*) |
| SUP.3 | | SUP.3.BP1 | Sviluppo strategia di quality assurance | |
| | | SUP.3.BP2 | Instaurazione standard di qualità | |
| | | SUP.3.BP3 | Definizione documentazione di qualità | |

Continua alla pagina successiva

**Tabella 5.1 – continua da pagina precedente**

| Processo base | Processo componente | Codice | Denominazione | Processo Mida Solutions |
|---|---|---|---|---|
| Processo di quality assurance | | SUP.3.BP4 | Assicurazione qualità su attività di processo | |
| | | SUP.3.BP5 | Assicurazione qualità sui prodotti | |
| | | SUP.3.BP6 | Documentazione dei risultati sulla qualità | |
| | | SUP.3.BP7 | Gestione delle deviazioni | |
| SUP.4 Processo di verifica | | SUP.4.BP1 | Sviluppo della strategia di verifica | PROD-PL |
| | | SUP.4.BP2 | Esecuzione verifica | PROD-PL |
| | | SUP.4.BP3 | Determinazione azioni per risultati verifica | PROD-PL |
| | | SUP.4.BP4 | Documentazione azioni per risultati verifica | |
| SUP.5 Processo di validazione | | SUP.5.BP1 | Sviluppo strategia di validazione | PROD-PL |
| | | SUP.5.BP2 | Esecuzione validazione | PROD-PL |
| | | SUP.5.BP3 | Determinazione azioni per risultati validazione | PROD-PL |
| | | SUP.5.BP4 | Documentazione azioni per risultati validazione | |
| SUP.6 Processo di revisione congiunta | | SUP.6.BP1 | Preparazione revisioni congiunte | |
| | | SUP.6.BP2 | Instaurazione criteri di revisione | |
| | | SUP.6.BP3 | Conduzione revisione congiunta di management | |
| | | SUP.6.BP4 | Conduzione revisione congiunta tecnica | |
| | | SUP.6.BP5 | Conduzione revisione congiunta di processo | |
| | | SUP.6.BP6 | Conduzione revisione congiunta di accettazione del sistema | |
| | | SUP.6.BP7 | Determinazione azioni per risulati revisione | |
| | | SUP.6.BP8 | Documentare azioni per risultati revisione | |
| SUP.7 Processo di audit | | SUP.7.BP1 | Sviluppo ed implementazione strategia di audit | |
| | | SUP.7.BP2 | Pianificazione di un audit | |
| | | SUP.7.BP3 | Esecuzione audit su attività di sviluppo sw | |
| | | SUP.7.BP4 | Esecuzione audit su attività di management | |
| | | SUP.7.BP5 | Esecuzione audit su performance di processo | |
| | | SUP.7.BP6 | Esecuzione audit su prodotti finali e sul sistema | |
| | | SUP.7.BP7 | Identificazione azioni correttive dai report di audit | |
| | | SUP.7.BP8 | Documentare azioni per report di audit | |
| SUP.8 Processo di risoluzione dei problemi | | SUP.8.BP1 | Instaurazione di un sistema per report problemi | |
| | | SUP.8.BP2 | Priorizzazione problemi | |
| | | SUP.8.BP3 | Determinazione azioni per i problemi | |
| | | SUP.8.BP4 | Documentazione azioni per i problemi | |
| | | SUP.8.BP5 | Revisione e distribuzione delle soluzioni | |
| | | SUP.8.BP6 | Analisi dei trend di problemi | |
| *Management* | | | | |
| MAN.1 Processo di management | | MAN.1.BP1 | Identificazione attività e task | PROD-PL/CUS-DEV-PL |
| | | MAN.1.BP2 | Valutazione di fattibilità su obiettivi processo | PROD-PL/CUS-DEV-PL |
| | | MAN.1.BP3 | Pianificazione e allocazione risorse ed infrastrutture | DEV |
| | | MAN.1.BP4 | Implementazione attività | DEV |
| | | MAN.1.BP5 | Controllo performance | DEV |
| | | MAN.1.BP6 | Revisione prodotti e valutazione risultati | PROD-MGMT |
| | | MAN.1.BP7 | Esecuzione azioni su deviazione performance | DEV |
| | | MAN.1.BP8 | Dimostrazione raggiungimento | DEV |
| MAN.2 Processo di project management | | MAN.2.BP1 | Definizione ambito lavorativo | PROD-PL |
| | | MAN.2.BP2 | Determinazione strategia di sviluppo | PROD-PL |
| | | MAN.2.BP3 | Selezione modello di ciclo di vita del software | PROD-PL |
| | | MAN.2.BP4 | Dimensionamento e stima di task e risorse | PROD-PL |
| | | MAN.2.BP5 | Sviluppo della Work Breakdown Structure | PROD-PL |
| | | MAN.2.BP6 | Identificazione requisiti su infrastrutture | PROD-PL |
| | | MAN.2.BP7 | Creazione schedulazione di progetto | PROD-PL |
| | | MAN.2.BP8 | Allocazione responsabilità | PROD-PL |
| | | MAN.2.BP9 | Identificazione interfacce | |
| | | MAN.2.BP10 | Instaurazionde ed implementazione piani di progetto | PROD-PL |
| | | MAN.2.BP11 | Documentazionde del progresso in confronto con i piani | DEV |
| | | MAN.2.BP12 | Correzione deviazioni | DEV |
| MAN.3 Processo di gestione qualità | | MAN.3.BP1 | Instaurazione degli obiettivi di qualità | |
| | | MAN.3.BP2 | Definizione della strategia complessiva | |
| | | MAN.3.BP3 | Identificazione delle attività di qualità | |
| | | MAN.3.BP4 | Esecuzione delle attività di qualità | |
| | | MAN.3.BP5 | Valutazione della qualità | |
| | | MAN.3.BP6 | Esecuzione di azioni correttive | |
| MAN.4 Processo di gestione rischi | | MAN.4.BP1 | Instaurazione ambito gestione rischi | PROD-PL |
| | | MAN.4.BP2 | Identificazione rischi | PROD-PL |
| | | MAN.4.BP3 | Analisi e creazione di priorità dei rischi | PROD-PL |
| | | MAN.4.BP4 | Definizione strategie di gestione rischi | PROD-PL |
| | | MAN.4.BP5 | Definizione metriche di rischio | PROD-PL |
| | | MAN.4.BP6 | Implementazione strategie di gestione rischi | PROD-PL |
| | | MAN.4.BP7 | Valutazione risultati strategie di gestione rischi | PROD-PL |
| | | MAN.4.BP8 | Esecuzione azioni correttive | PROD-PL |
| *Organization* | | | | |
| ORG.1 Processo di allineamento organizzativo | | ORG.1.BP1 | Sviluppo visione strategica | (*) |
| | | ORG.1.BP2 | Condivisione visione | (*) |
| | | ORG.1.BP3 | Sviluppo di una cultura di qualità | (*) |
| | | ORG.1.BP4 | Costituzione gruppi autorizzati | (*) |

Continua alla pagina successiva

**Tabella 5.1 – continua da pagina precedente**

| Processo base | Processo componente | Codice | Denominazione | Processo Mida Solutions |
|---|---|---|---|---|
| | | ORG.1.BP5 | Incentivazione | (*) |
| ORG.2 Proc. di miglioramento | Costituzione dei processi | ORG.2.1.BP1 | Definizione degli obiettivi | |
| | | ORG.2.1.BP2 | Identificazione attività, ruoli, autorità e responsabilità | |
| | | ORG.2.1.BP3 | Definizione e documentazione processi implementati nell'organizzazione | |
| | | ORG.2.1.BP4 | Costituzione politiche organizzative | |
| | | ORG.2.1.BP5 | Instaurazione aspettativa di performance | |
| | | ORG.2.1.BP6 | Sviluppo processo | |
| | | ORG.2.1.BP7 | Verifica sviluppo processi standard | |
| | | ORG.2.1.BP8 | Cattura dati di processo | |
| | | ORG.2.1.BP9 | Manutenzione processi standard | |
| | Valutazione dei processi | ORG.2.2.BP1 | Determinazione metodo di valutazione | |
| | | ORG.2.2.BP2 | Definizione obiettivi valutazione | |
| | | ORG.2.2.BP3 | Definizione input di valutazione | |
| | | ORG.2.2.BP4 | Pianificazione valutazione | |
| | | ORG.2.2.BP5 | Esecuzione valutazione per la collezione dati | |
| | | ORG.2.2.BP6 | Validazione dati | |
| | | ORG.2.2.BP7 | Identificazione punti di forza e di debolezza | |
| | | ORG.2.2.BP8 | Manutenzione dei risultati di valutazione | |
| | | ORG.2.2.BP9 | Valutazione risultati | |
| | | ORG.2.2.BP10 | Condivisione risultati di valutazione | |
| | Miglioramento dei processi | ORG.2.3.BP1 | Identificazione opportunità di miglioramento | |
| | | ORG.2.3.BP2 | Definizione obiettivi e attività di miglioramento | |
| | | ORG.2.3.BP3 | Studio del processo | |
| | | ORG.2.3.BP4 | Identificazione miglioramenti | |
| | | ORG.2.3.BP5 | Creazione priorità miglioramenti | |
| | | ORG.2.3.BP6 | Definizione misure di impatto | |
| | | ORG.2.3.BP7 | Cambiamento del processo | |
| | | ORG.2.3.BP8 | Conferma miglioramenti | |
| | | ORG.2.3.BP9 | Condivisione del processo | |
| ORG.3 Processo di gestione delle risorse umane | | ORG.3.BP1 | Identificazione risorse umane necessarie | (*) |
| | | ORG.3.BP2 | Sviluppo o acquisizione di formazione | (*) |
| | | ORG.3.BP3 | Formazione personale | (*) |
| | | ORG.3.BP4 | Reclutamento di staff qualificato | (*) |
| | | ORG.3.BP5 | Definizione criteri di valutazione | (*) |
| | | ORG.3.BP6 | Valutazione performance dello staff | (*) |
| | | ORG.3.BP7 | Fornitura feedback di valutazione | (*) |
| | | ORG.3.BP8 | Manutenzione documentazione di staff | (*) |
| | | ORG.3.BP9 | Definizione team di progetto | (*) |
| | | ORG.3.BP10 | Abilitazione team di progetto | (*) |
| | | ORG.3.BP11 | Manutenzione interazioni tra team di progetto | (*) |
| ORG.4 Processo di infrastrutture | | ORG.4.BP1 | Identificazione requisiti di ingegnerizzazione software dell'ambiente | |
| | | ORG.4.BP2 | Fornitura ambiente di ingegnerizzazione | |
| | | ORG.4.BP3 | Fornitura supporto per utilizzo dell'infrastruttura | |
| | | ORG.4.BP4 | Manutenzione ambiente di ingegnerizzazione sw | |
| | | ORG.4.BP5 | Fornitura spazio di lavoro adeguato | |
| | | ORG.4.BP6 | Assicurazione integrità e sicurezza dei dati | |
| | | ORG.4.BP7 | Fornitura strumenti di accesso remoto | |
| ORG.5 Processo di misura | | ORG.5.BP1 | Instaurazione metriche per gestione processi | |
| | | ORG.5.BP2 | Instaurazione metriche per qualità prodotti | |
| | | ORG.5.BP3 | Conduzione gestione quantitativa dei processi | |
| | | ORG.5.BP4 | Misurazione qualità dei prodotti lavorativi | |
| | | ORG.5.BP5 | Esecuzione misurazioni su dati disponibili per le decisioni | |
| | | ORG.5.BP6 | Definizione dei benchmark | |
| | | ORG.5.BP7 | Esecuzione del benchmark dei processi | |
| ORG.6 Processo di riutilizzo | | ORG.6.BP1 | Definizione strat. organizzativa di riutilizzo | |
| | | ORG.6.BP2 | Creazione libreria di riutilizzo | (*) |
| | | ORG.6.BP3 | Identificazione entità riutilizzabili | |
| | | ORG.6.BP4 | Sviluppo entità riutilizzabili | |
| | | ORG.6.BP5 | Mantenimento delle entità riutilizzabili | |
| | | ORG.6.BP6 | Documentazione e certificazione delle entità riutilizzabili | |
| | | ORG.6.BP7 | Informazione dei potenziali utenti | |

## 5.3 Criteri di valutazione e risultati

### 5.3.1 I criteri di valutazione forniti dal modello

Il modello definisce una scala di misurazione per le capacità di un qualsiasi processo. All'interno della dimensione di capacità del modello di riferimento, la misura è basata su un set di **Attributi di Processo** (PA) utilizzati per determinare se un processo ha raggiunto una data capacità. Ogni attributo misura un particolare aspetto di una capacità di processo, gli stessi attributi sono misurati in una scala percentuale e quindi forniscono una visione molto più dettagliata delle capacità del processo.

Di seguito vengono elencati gli attributi di processo inerenti ad ogni livello di capacità di processo.

- **Livello 0: Processo Incompleto**
  Il processo non è implementato oppure non raggiunge gli output prefissati. A questo livello non esiste evidenza di alcun raggiungimento sistematico di alcun attributo di processo.

- **Livello 1: Processo Eseguito**
  L'implementazione del processo ottiene gli output prefissati. I seguenti attributi dimostrano il raggiungimento di questo livello:

  - **PA 1.1 Attributo di performance di processo** - Il grado con cui il processo raggiunge i propri risultati trasformando input identificabili in prodotti lavorativi.

- **Livello 2: Processo Gestito**
  Il processo eseguito a livello precedente ora è attuato in una modalità gestita (pianificata, documentata, verificata e adeguata) basata su obiettivi predefiniti.

  - **PA 2.1 Attributo di gestione delle performance** - Il grado con cui le performance del processo sono gestite al fine di ottenere prodotti che raggiungano obiettivi predefiniti.
  - **PA 2.2 Attributo di gestione dei prodotti lavorarivi** - Il grado con cui le performance del processo sono gestite al fine di ottenere prodotti lavorativi propriamente documentati, controllati e verificati.

- **Livello 3: Processo Definito**
  Il processo gestito a livello precedente ora è eseguito utilizzando un processo definito basato su principi di software engineering e capace di ottenere gli output predefiniti.

  - **PA 3.1 Attributo di definizione di processo** - Il grado con cui le performance di un processo utilizzano una definizione di processo basata su standard per ottenere gli output predefiniti.

  - **PA 3.2 Attributo di risorsa di processo** - Il grado con cui il processo utilizza un adeguato numero di risorse (umane e infrastrutturali) propriamente allocate.

- **Livello 4: Processo Prevedibile**
  Il processo precedentemente definito ora è eseguito ripetutamente all'interno di limiti predefiniti.

  - **PA 4.1 Attributo di misurabilità** - Il grado con cui le misure sui prodotti e obiettivi vengono utilizzati per assicurare che le performance dei processi supportino il raggiungimento di obiettivi predefiniti in linea con gli obiettivi di business.
  - **PA 4.2 Attributo di controllo di processo** - Il grado con cui il processo è controllato tramite la collezione, analisi, ed utilizzo di prodotti e processi di misura al fine di correggere, quando necessario, le performance del processo per raggiungere obiettivi predefiniti.

- **Livello 5: Processo Ottimizzato**
  Il processo precedentemente prevedibile ora cambia dinamicamente per adattarsi efficacemente agli obiettivi di business attuali e futuri.

  - **PA 5.1 Attributo di cambiamento di processo** - Il grado con cui i cambiamenti nella definizione, gestione e performance del processo vengono controllati per raggiungere gli obiettivi di business dell'organizzazione.
  - **PA 5.2 Attributo di miglioramento continuo** - Il grado con cui i cambiamenti al processo sono identificati ed implementati per assicurare il miglioramento continuo nel raggiungimento degli obiettivi di business dell'organizzazione.

La scala di valutazione è una percentuale che rappresenta il grado di raggiungimento dell'attributo. La scala viene utilizzata per calibrare i livelli di raggiungimento di una data capacità di un attributo di processo:

- **N** Non raggiunto:
  0% - 15% - Non c'è alcuna o esiste una piccola evidenza del raggiungimento dell' attributo nel processo valutato.

- **P** Parzialmente raggiunto:
  16% - 50% - Esiste evidenza di un approccio sistematico e del raggiungimento dell'attributo. Alcuni aspetti del raggiungimento potrebbero essere non prevedibili.

- **L** Largamente raggiunto:
  51% - 85% - Esiste evidenza di un approccio sistematico e di un significativo raggiungimento dell'attributo. Le performance del processo potrebbero variare in alcune aree o in alcune unità lavorative.

| Scala | Attributo di processo | Valutazione |
|---|---|---|
| **Livello 1** | Performance di processo | L o F |
| **Livello 2** | Performance di processo | F |
| | Gestione delle performance | L o F |
| | Gestione dei prodotti lavorativi | L o F |
| **Livello 3** | Performance di processo | F |
| | Gestione delle performance | F |
| | Gestione dei prodotti lavorativi | F |
| | Definizione e assestamento del processo | L o F |
| | Risorse del processo | L o F |
| **Livello 4** | Performance di processo | F |
| | Gestione delle performance | F |
| | Gestione dei prodotti lavorativi | F |
| | Definizione e assestamento del processo | F |
| | Risorse del processo | F |
| | Misurazioni del processo | L o F |
| | Controllo del processo | L o F |
| **Livello 5** | Performance di processo | F |
| | Gestione delle performance | F |
| | Gestione dei prodotti lavorativi | F |
| | Definizione e assestamento del processo | F |
| | Risorse del processo | F |
| | Misurazioni del processo | F |
| | Controllo del processo | F |
| | Cambiamento del processo | L o F |
| | Miglioramento continuo | L o F |

**Tabella 5.2.** Valutazioni dei livelli di capacità.

- **F** Completamente raggiunto: 86% - 100% - Esiste evidenza di un approccio completo e sistematico e di un pieno ottenimento dell'attributo. Non esistono debolezze significative nell'unità organizzativa valutata.

La tabella 5.2 riporta i criteri secondo cui valutare il livello di capacità del processo produttivo.

### 5.3.2 Esito della valutazione

Valutando i risultati della tracciatura degli elementi forniti dal modello sui processi aziendali di Mida Solutions raccolti nella sezione 5.2, sono state assegnate le valutazioni sugli attributi del processo produttivo riportate nella tabella 5.3. In base ai giudizi assegnati e ai criteri di valutazione dei livelli di capacitá (tabella 5.2), appare chiaro come il processo produttivo di Mida rientri nella categoria

| ID | Attributo di processo | Valutazione |
|---|---|---|
| PA 1.1 | Performance di processo | F |
| PA 2.1 | Gestione delle performance | F |
| PA 2.2 | Gestione dei prodotti lavorativi | L |
| PA 3.1 | Definizione di processo | P |
| PA 3.2 | Risorsa di processo | P |
| PA 4.1 | Misurabilità | N/P |
| PA 4.2 | Controllo del processo | N |
| PA 5.1 | Cambiamento di processo | N |
| PA 5.1 | Miglioramento continuo | N |

**Tabella 5.3.** Valutazioni assegnate ai diversi attributi di processo.

***Livello 2 - Gestito***. La mappatura ha messo in evidenza una buona e completa definizione dei processi delle categorie *Engineering* e *Management*, il che indica la presenza di una buona gestione della produzione. Al contrario, i fattori che non permettono il raggiungimento del livello successivo consistono essenzialmente nella solo parziale definizione di alcuni processi, presenti invece nel modello di riferimento, e nella mancanza di documentazione che certifichi la politica di allocazione delle risorse, siano esse umane o infrastrutturali. La mancanza dell'implementazione di numerosi processi delle categorie *Support* e *Organization* impediscono, per il momento, il raggiungimento dei livelli 4 e 5.

# Parte II

# Il progetto WQF

# Capitolo 6

# Il progetto WQF

La seconda parte di questo lavoro è stata dedicata allo sviluppo di un'analisi dei requisiti e alla conseguente implementazione di un software per il sistema di gestione della qualità (SGQ). Anche in questo caso, il lavoro è iniziato con riferimento alla norma ISO:9001 ed è, poi, proseguito verso aspetti più pratici, legati, principalmente alle esigenze espresse dai rappresentanti dell'azienda. La disponibilità di software (commerciali o meno) per SGQ non è sufficiente ad affermare che essi si possano adottare in tutte le realtà aziendali. Nella maggior parte dei casi si è osservato che questi software sono stati progettati e sviluppati avendo come riferimento una *generica* realtà; per questo motivo, tali tools, sebbene ricchi di funzionalità, non possono essere applicati all'ambito di interesse di questo lavoro. In particolare, le analisi effettuate hanno rilevato che non è possibile utilizzare software commerciali senza dover rivedere le esigenze aziendali, non potendo, quindi, sfruttare appieno le caratteristiche che si desiderano riguardo al software di SGQ.

Onde evitare, quindi, che l'introduzione di un sistema per la qualità imponga una *sovrastruttura* aziendale, si è deciso di intraprendere un percorso, sicuramente più ostico, ma anche più flessibile, di sviluppo *ad hoc* del software di ausilio al suddetto sistema.

## 6.1 Gli obbiettivi

Le funzioni richieste dal sistema sono riunite come segue. Lo strumento da analizzare deve:

- permettere di disegnare i processi aziendali e memorizzarne le istanze create;
- tenere traccia dei dati prodotti;
- produrre in la documentazione per il manuale della qualità nel momento in cui questa venga richiesta;

- mantenere lo storico delle revisioni dei processi, per poter sempre verificare quale versione di un processo fosse attiva in un certo momento;

- mantenere lo storico documenti prodotti in passato;

- fornire un motore di workflow, ovvero uno strumento che, seguendo passo passo i grafi dei processi, guidi le attività del personale in modo da aiutarlo ad eseguile rispettando le procedure aziendali. In riferimento all'esecuzione dei processi, il sistema deve permettere di mantenere informazioni sulle attività svolte, in modo da poterle tracciare con semplicità. In linea di principio il funzionamento di quest'ultimo è piuttosto semplice: un processo è composto da varie attività di tipologie differenti (ad es. start, finish, scelta ). Ad ognuna di queste attività viene associata una vista per l'utente in funzione delle caratteristiche dell'attività stessa. Eseguire un processo significa crearne un istanza attivando, quindi, il corrispondente nodo start, che genererà la pagina corrispondente all'attivita immediatamente successiva. Una volta conclusa ogni attività il sistema si occuperà di fornire a chi di dovere la vista dell'attività che segue;

- gestire gli accessi degli utenti a dati e attività, in base al ruolo di appartenenza all'interno dell'azienda.

È risultato chiaro, sin dai primi colloqui con l'azienda, che lo sforzo nella realizzazione del software debba essere messo non tanto nella generalità dei moduli e delle funzionalità (caratteristica già presente nei software commerciali), quanto nella capacità del sistema di mantenere tutti i dati ed i documenti necessari al SGQ. Il software deve essere, infatti, capace di descrivere la realtà aziendale (intesa come insieme di processi, procedure, ecc.) anche a fronte di cambiamenti repentini e di riuscire a tener traccia dei cambiamenti avvenuti nel tempo in modo semplice ed efficace.

Come già detto, uno degli aspetti chiave di un software per SGQ è la gestione della documentazione. Questo aspetto risulta tanto importante per l'ottenimento della certificazione, quanto spesso critico per le varie figure aziendali. La produzione dei documenti del SGQ è spesso vista dal personale come un inutile (o peggio dannoso) aumento del carico di lavoro. Per questi motivi spesso ci si scontra con realtà che si sentono eccessivamente vincolate dalla gestione della documentazione e, in alcuni casi, questa gestione viene posticipata fintantochè non risulti improrogabile. Il risultato effettivo di un tale approccio è la presenza di incoerenze nei dati presentati all'eventuale ispettore, con il rischio di ricevere un ammonimento o, in casi estremi, la revoca della certificazione. Spesso una gestione non corretta della documentazione non può essere attribuita unicamente al personale aziendale: in molti casi, infatti, l'aumento del carico di lavoro non è attribuibile unicamente alle prescrizioni normative, ma anche al tipo di

ausilio (ad esempio il software) che si utilizza per la gestione degli aspetti relativi alla qualità. Questa affermazione risulta tanto più verosimile se si considera una realtà aziendale in cui si utilizzano software commerciali per il SGQ. Spesso questi pacchetti "impongono" un modo di lavorare e, di conseguenza, di produrre documenti, che non risulta coerente con la prassi quotidiana dell'azienda. Nella fase di progettazione di un nuovo software per il sistema di gestione della qualità, questo è un aspetto fondamentale e va, perciò, analizzato in dettaglio. Il software da realizzare deve, quindi, avere la capacità di effettuare quella che viene chiamata *autodocumentazione*, ovvero deve essere in grado di produrre in modo automatico documenti e registrazioni coerenti con le specifiche normative. L'espressione "in modo automatico" sta a significare che il carico di lavoro svolto dal personale dipendente deve essere minimo.
Per meglio comprendere come sia possibile sviluppare un software in grado di presentare una forte proprietà di autodocumentazione, si deve prima di tutto capire come i documenti debbano essere redatti, promulgati, revisionati e distrutti in modo da ottemperare ai requisiti normativi. Uno dei concetti fondamentali espressi dalla norma è la *rintracciabilità* dei documenti. Ciò significa che ogni singola registrazione deve essere sempre accessibile in modo immediato e definito. Devono essere, quindi, note in ogni momento la collocazione del documento, la versione e/o revisione e l'ambito di applicabilità (chi deve conoscere il documento, chi può leggerlo, ...). I principali problemi relativi alla realizzazione di un'architettura software in grado di implementare queste funzionalità risiedono nel riuscire a rendere il meccanismo di rintracciabilità trasparente ed efficace. È, infatti, inutile prevedere delle procedure atte alla redazione e alla pubblicazione di documenti se, successivamente, la loro consultazione risulta complicata. L'ovvia conseguenza sarebbe quella di produrre documenti che non vengono mai consultati, oppure di scoraggiare la produzione dei documenti stessi. Un risultato frequente consiste nel produrre documentazione che presenta delle carenze (mancano i documenti al momento della visita ispettiva) o delle incongruenze (i documenti prescrivevano una certa procedura che, però, non è stata eseguita in quanto il documento non è stato consultato).

# Capitolo 7

# Definizione del piano di lavoro

La realizzazione di un *progetto*[1] ne prevede l'attuazione in un tempo relativamente limitato, e comunque ben definito, che può andare da qualche settimana a un paio di anni; qualora, invece, il tempo richiesto per il raggiungimento dell'obiettivo sia superiore, è allora più corretto parlare di *programma*, con obiettivi più complessi e di più ampia portata rispetto a quelli di un progetto che, nel loro insieme, costituiscono generalmente realizzazioni di tipo strategico e con effetti di lungo termine.

La definizione dettagliata di un modello metodologico di riferimento è essenziale per costituire una struttura ordinata e standardizzata, nell'ambito di un'organizzazione, che permetta, fra l'altro, la definizione degli aspetti organizzativi, gestionali e dei documenti da produrre per l'alimentazione del *database di progetto*. In generale, si definisce *ciclo di vita* di un progetto l'insieme delle fasi che, succedendosi nel tempo, danno luogo alla realizzazione di quanto previsto dal progetto; un esempio di fasi relative a un progetto informatico può essere scomposto è:

- definizione dei requisiti utente;
- stesura delle specifiche funzionali;
- disegno di primo livello;
- disegno dettagliato;
- produzione e integrazione;
- test di sistema;
- test di accettazione;

[1] Una definizione generale di progetto è quella di un insieme di attività finalizzate al raggiungimento di un determinato obiettivo, univocamente definito, attraverso l'impiego di risorse umane, materiali, tecnologiche, temporali e finanziarie, nel rispetto di prefissati vincoli in termini di tempi, costi e qualità.

- operatività e manutenzione.

In questo contesto, per *controllo* si intende un adeguato governo del processo realizzativo: il Project Manager[2] ha, in generale, la responsabilità sia del "prodotto" che del "processo di produzione", il cui controllo può essere attuato tramite idonea attività di pianificazione e reporting. Il controllo viene, infatti, assicurato tramite la pianificazione delle attività e il monitoraggio periodico del relativo stato di avanzamento, al fine di individuare possibili azioni correttive atte a recuperare eventuali scostamenti dagli obiettivi attesi ovvero a effettuare una ripianificazione. Nella pianificazione, d'altra parte, occorre tener conto che nessun piano è in grado di focalizzare a priori tutti i problemi che verranno incontrati o il tempo preciso che verrà richiesto da ogni singola attività: per questo è previsto che il piano venga periodicamente riemesso, e può costituire uno sforzo inutile scendere nella sua definizione ad un livello di dettaglio troppo spinto. La prima emissione del Piano di Progetto dovrebbe, tuttavia, essere sufficientemente valida e i tempi e i costi totali pianificati non dovrebbero, in genere, subire variazioni molto significative, se non in casi eccezionali ed opportunamente motivabili e documentabili. Una buona tecnica di gestione del controllo è la scomposizione del progetto in attività di complessità inferiore; un esame più approfondito può rilevare che il sistema da realizzare può essere opportunamente segmentato e, per progetti di dimensioni rilevanti, se ne possono in tal modo ridurre i rischi.

In dipendenza delle dimensioni e della durata temporale di un progetto, l'insieme delle attività che lo costituiscono e che concorrono alla sua definizione possono essere, quindi, scomposte in attività di livello gerarchicamente inferiore, più semplici e quindi più facilmente definibili e controllabili sul piano realizzativo: quanto più ciascuna attività risulta chiaramente definita nei suoi aspetti tecnici, temporali ed economici tanto più sarà semplice valutarne l'evoluzione progettuale e l'eventuale scostamento dei suoi parametri di valutazione dai valori attesi. Una delle più importanti fasi nella definizione di un progetto è, quindi, l'identificazione e la descrizione di ciascuna attività che lo costituisce e l'ulteriore suddivisione in attività più semplici, fino ad un livello tale per cui ciascuna attività elementare, o compito, è chiaramente ed univocamente identificata, in termini di prodotto da realizzare e di tempi e costi di esecuzione. L'attività di individuazione delle componenti elementari deriva dalla necessità di disporre di un procedimento ordinato e sistematico per una definizione che assicuri una corretta interrelazione fra tutti gli elementi, attraverso la creazione di una *struttura analitica di progetto* o *struttura di scomposizione del lavoro*. La struttura che ne deriva è detta *Work Breakdown Structure* (WBS) o *Project Breakdown Structure* (PBS) e costituisce una rappresentazione del progetto, in forma grafica e/o descrittiva che, suddividendo le attivitá in livelli, consente un'analisi di dettaglio indispensabile per una corretta identificazione delle attività elementari la cui ese-

[2]Il Project Manager è la figura centrale e di riferimento per qualunque progetto che riveste il duplice ruolo di responsabile unico del buon esito del progetto e punto riferimento nei confronti del committente, del management aziendale e dell'intero team di progetto.

cuzione integrata conduce alla realizzazione dell'intero progetto. Una struttura WBS consente di evidenziare gli elementi oggetto di consegna al cliente (detti anche *deliverables*) e i principali compiti funzionali da eseguire per la realizzazione di ciascuno di essi, attraverso la definizione e l'esecuzione di test, intesi come risultati materiali o immateriali delle attività di progetto. É bene evidenziare che la struttura di una WBS non ha alcuna relazione con lo sviluppo temporale delle attività di progetto, mentre è la successiva attività di pianificazione a dover ordinare cronologicamente fasi e compiti; le caratteristiche fondamentali della metodologia derivate dall'impiego di una struttura WBS sono:

- il collegamento fra loro e al prodotto finale i singoli compiti;
- la scomposizione dell'elemento di più alto livello (progetto) nei principali elementi costitutivi: sistemi, facilities (risorse ed accessori), deliverables;
- la scomposizione di ciascun elemento costitutivo in oggetti di entità più semplici (in termini di entità e complessità) e costo inferiore, fino all'identificazione di un oggetto ben definito da consegnare;
- la visualizzazione del progetto nella sua interezza, evidenziando la sua complessità e i collegamenti fra i vari elementi.

Dalla costruzione della WBS, scaturisce l'individuazione dei principali compiti da eseguirsi dalle singole funzioni per la realizzazione dell'oggetto: la scomposizione di un progetto in una Work Breakdown Structure, infatti, consente di identificare elementi e compiti chiaramente gestibili e attribuibili a specifiche responsabilità, la cui attuazione possa essere pianificata, valutata, schedulata e controllata: in tal modo il progetto viene definito in ogni sua parte e consente la realizzazione di riepiloghi sulla realizzazione del progetto stesso. Una WBS, pertanto, costituisce un valido elemento di comunicazione, evolvendo e riflettendo i piani correnti in base al reale sviluppo del progetto ed esplicitando maggiori dettagli in vista dell'esecuzione del progetto; costituisce, inoltre, un efficace ausilio per individuare tutti gli elementi fondamentali per il buon esito del progetto, evitando di trascurare aspetti meno evidenti e permettendo di chiarire ruoli e impegni del responsabile dell'esecuzione di ciascun compito. L'analisi di una struttura WBS deve essere condotta con tutti gli attori coinvolti nella realizzazione del progetto, allo scopo di raggiungere alla piena condivisione della sua validità, identificando e condividendo la strutturazione in work packages (compiti e connesse attività per il loro completamento) che devono essere opportunamente pianificati, valutati, budgetati, schedulati e controllati.

Per quanto riguarda il progetto WQF descritto in questa tesi, le attività preliminari per la definizione del progetto sono state effettuate il 10 settembre 2009 (vedi figura 7.1) dal team di sviluppo sotto la supervisione del Prof. Bertocco e dell'Ing. Mauro Franchin in rappresentanza di Mida Solutions. La tecnica utilizzata per la definizione delle attività di progetto, e anche quella maggiormente

consigliata, è stata una seduta di brainstorming. L'utilizzo di post-it sui quali scrivere le varie proposte risulta molto utile in quanto permette di modificare frequentemente la loro disposizione, sia nella definizione della WBS, sia nella specifica dei vincoli di precedenza.

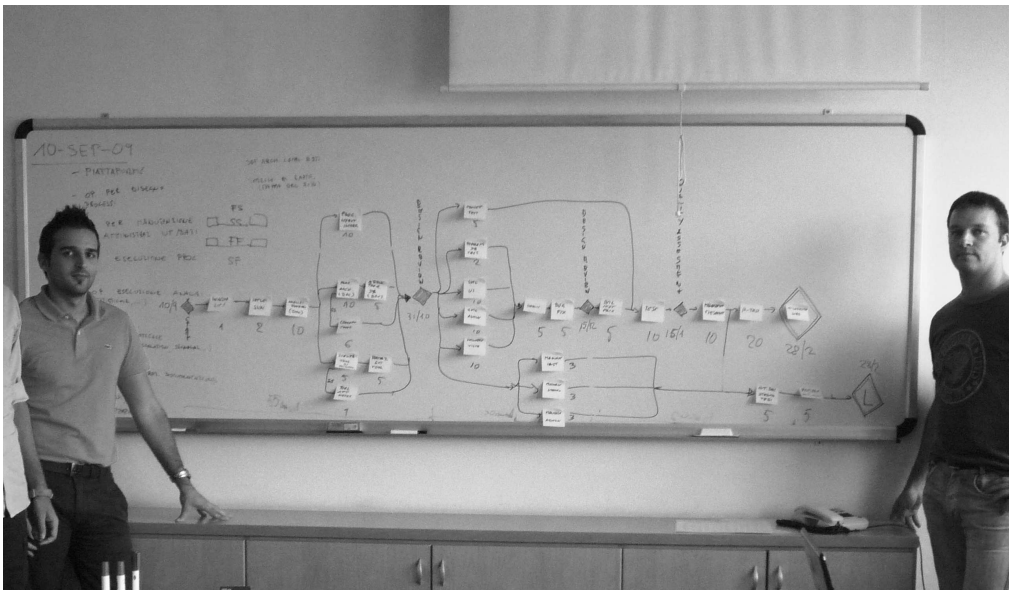


**Figura 7.1.** 10 settembre 2009: identificazione delle attività di progetto, definizione delle durate temporali e determinazione dell'ordine di precedenza.

Il diagramma di Gantt costituisce un efficace strumento di ausilio alla pianificazione e consente di disporre in forma integrata e visuale, di semplice e immediata comprensibilità, l'evoluzione temporale complessiva del progetto e l'interrelazione fra tutte, o le principali, attività in cui il progetto stesso è stato scomposto. Un diagramma di Gantt è costituito da un elenco delle principali attività e delle sottoattività, generalmente coincidenti con quelle definite nella costruzione della WBS, disposte in forma tabellare e collegate temporalmente in coerenza con con quanto definito in precedenza. In un diagramma di Gantt, per ciascuna attività possono essere specificate varie informazioni che la caratterizzano; generalmente è specificato un identificativo univoco, la data di inizio e di conclusione, la durata prevista, le risorse allocate, le attività da cui questa dipende e quelle ad essa subordinate. La figura 7.2 mostra il diagramma di Gantt per il progetto WQF.

Quanto più la pianificazione del un progetto è effettuata in modo dettagliato ed integrato a livello di attività elementari, tanto più preciso risulta il controllo della sua evoluzione; in conseguenza di ciò, la possibilità di assicurare il buon esito di un progetto, ovvero di minimizzare gli scostamenti dei risultati rispetto agli obiettivi attesi, può conseguirsi attraverso una schedulazione dettagliata

fino al livello del singolo compito ed opportunamente integrata. La tecniche introdotte in precedenza, se correttamente applicate, possono agevolare il raggiungimento di tale obiettivo, in quanto è in grado di assicurare un controllo puntuale ed istantaneo dello svolgimento di ogni attività, attraverso i compiti elementari in cui è stata scomposta, e di verificare, tramite l'interazione fra le attività e i compiti, gli impatti che uno slittamento temporale sul completamento di un'attività possono indurre sul progetto o su sue singole parti. D'altro canto, nella scomposizione delle attività e, quindi, nella definizione dei reticoli, occorre tener conto che se questi risultano troppo dettagliati, il reticolo integrato del progetto può divenire complesso, di difficile immediatezza descrittiva e richiedere, inoltre, strumenti elettronici di ausilio potenti, in grado di elaborare i dati e fornire risposte in tempi rapidi. Occorre, pertanto, individuare una soluzione di compromesso nella scomposizione delle attività, perché se da un lato quanto più questa è dettagliata più risulta semplice assicurare il controllo puntuale di ogni singola attività, dall'altro una eccessiva frammentazione può renderne più complessa e onerosa la gestione, complicando oltremodo l'analisi dell'evoluzione del progetto e l'eventuale esecuzione di simulazioni a supporto della pianificazione e delle scelte progettuale.

Per garantire il buon esito del progetto, in riferimento ai vincoli da rispettare, è necessario mantenere sotto costante controllo gli indicatori di tempi, costi e qualità precisati in fase di definizione e pianificazione. Tale attività di controllo costituisce uno dei momenti più critici dell'intero processo di Project Management in quanto dall'osservazione puntuale degli indicatori citati è possibile intraprendere eventuali azioni correttive ovvero procedere a una ripianificazione del progetto, in parte o in toto, per raggiungere l'obiettivo prefissato e per minimizzare l'eventuale scostamento dovuto ad iniziali errori di valutazione o fenomeni intervenuti successivamente, che possano compromettere il buon esito dell'intero progetto o dar luogo al mancato rispetto dei vincoli inizialmente prefissati. In particolare, per quanto concerne il progetto WQF, sono stati svolti incontri periodici tra i componenti del team di sviluppo per verificare lo stato di avanzamento delle attività e per rivedere, se necessario, il piano di lavoro.

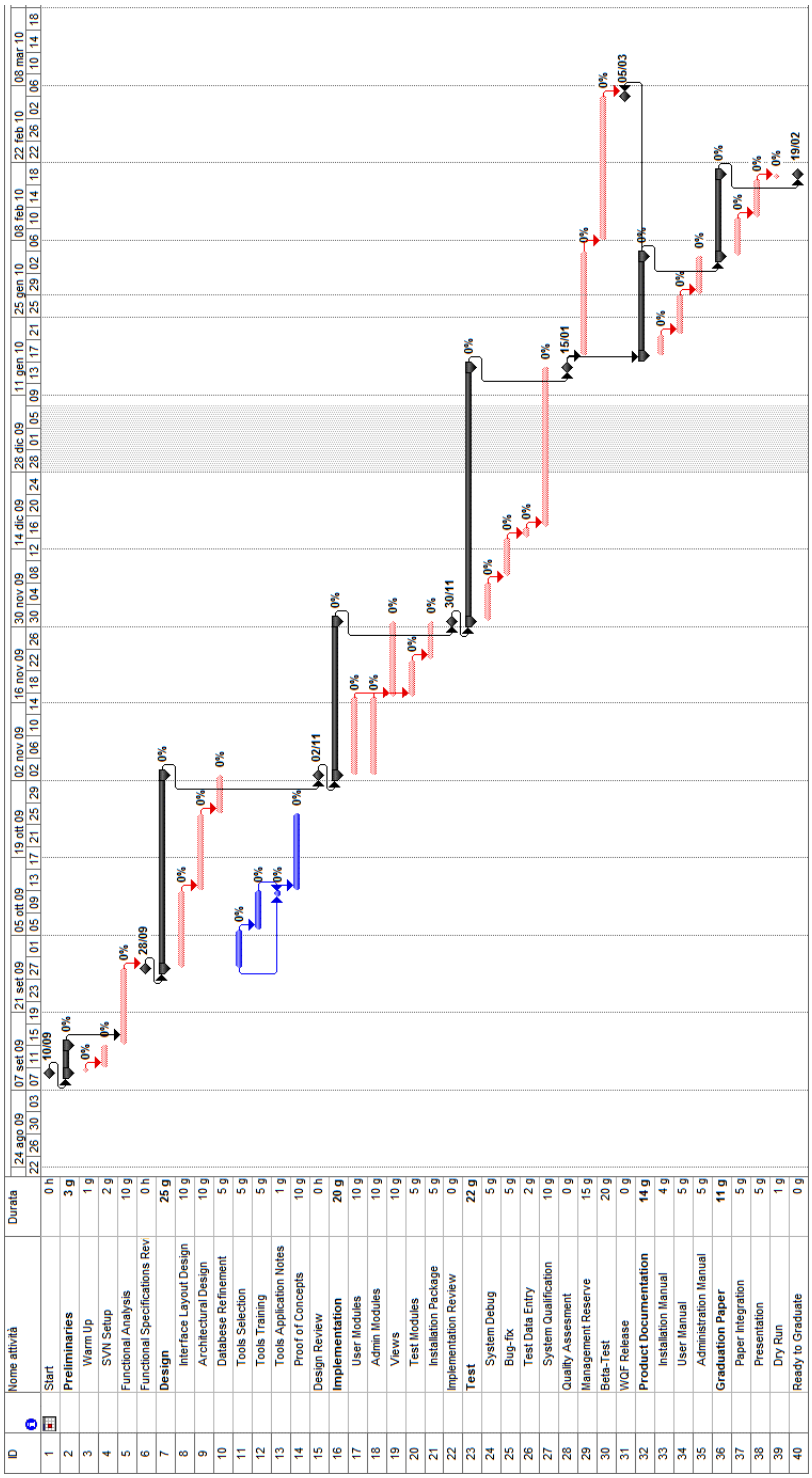


**Figura 7.2.** Diagramma di Gantt del progetto WQF.

# Capitolo 8

# Analisi funzionale

Comprendere i requisiti di un problema è uno fra i compiti più difficili per un ingegnere del software. La prima volta che si sente parlare di *analisi dei requisiti*, nota anche come *analisi funzionale*, non sembra un'attività tanto complicata. La prima cosa che viene in mente è che il cliente dovrebbe sapere ciò di cui ha bisogno. L'utente finale dovrebbe avere una discreta conoscenza delle caratteristiche e delle funzioni che forniscono dei benefici. Sorprendentemente, in molti casi, ciò non accade. E anche se i clienti e gli utenti finali sono espliciti quanto a comunicare i loro bisogni, tali bisogni cambieranno certamente durante lo sviluppo del progetto. Dunque, l'ingegneria dei requisiti non è un'attività tanto semplice.

Si cerca, ora, di definire con maggiore chiarezza di cosa si tratta. L'ingegneria dei requisiti aiuta gli ingegneri software a capire meglio il problema che stanno cercando di risolvere. Comprende un sistema di compiti che conducono alla comprensione dell'impatto del software, di ciò che i clienti desiderano e del modo in cui gli utenti finali interagiranno con il software. Questa attività deve essere svolta dagli ingegneri del software (chiamati anche ingegneri di sistemi o analisti) e dalle altre figure coinvolte nello sviluppo del progetto (manager, clienti, utenti finali, ecc.). La progettazione e la realizzazione di un elegante applicativo in grado di risolvere un problema errato non è di alcuna utilità. Questo è il motivo fondamentale per cui è importante sapere ciò che il cliente desidera prima di iniziare a progettare e costruire un sistema. Lo scopo del processo di analisi funzionale è quello di fornire a tutte le parti coinvolte una documentazione scritta del problema. Ciò può essere ottenuto in vari modi: utilizzando scenari utente, funzioni ed elenchi di funzioni caratteristiche, modelli analitici o specifiche.

Come detto precedentemente, la progettazione e costruzione di un software è un processo difficile. La realizzazione di software è così coinvolgente che molti sviluppatori spesso iniziano a realizzarlo ancor prima di avere una chiara idea di quanto sia necessario per procedere. Essi sostengono che le cose diverranno più chiare a mano a mano che si realizzerà il progetto; che tutte le figure coinvolte nel progetto potranno comprendere meglio i propri bisogni solo dopo aver esa-

minato le prime iterazioni del software; che le cose cambiano così rapidamente che l'analisi dei requisiti è una perdita di tempo; che quello che conta è produrre un programma funzionante e che tutto il resto è secondario. Ciò che rende così intriganti questi argomenti è che contengono alcuni elementi di verità[1], ma si tratta di un grave errore che può comportare il fallimento del progetto software.

> "*La parte più complessa nella realizzazione di un sistema software consiste nel decidere cosa realizzare. Nessuna parte del lavoro pregiudica maggiormente il risultato se viene eseguita in modo errato. Nessuna altra parte è più difficile da correggere successivamente.*"
>
> Fred Brooks[2]

L'analisi dei requisiti, come ogni altra attività dell'ingegneria del software, deve, quindi, essere adattata ai bisogni del processo, del progetto, del prodotto e delle persone che svolgono il lavoro. In alcuni casi si può scegliere un approccio abbreviato. In altri, invece, si deve svolgere in maniera rigorosa ogni singola operazione definita nell'ambito di un'ingegneria completa dei requisiti. In generale, il team di sviluppo software deve adattare il proprio approccio all'analisi dei requisiti. Adattarsi non deve essere un sinonimo di arrendersi. È fondamentale che il team di sviluppo software faccia uno sforzo reale per comprendere i requisiti di un problema *prima* di tentare di risolverlo. Non intendo essere prolisso per non annoiare il lettore. Questo paragrafo deve servire ad introdurre alcuni concetti importanti di ingegneria del software utili alla comprensione del lavoro svolto e non per emulare un libro di testo.

## 8.1 L'analisi del sistema

Nell'approfondire le diverse funzionalità del sistema illustate nei prossimi paragrafi, sono stati fatti esempi riferiti ad una probabile architettura che vede i dati memorizzati in una base di dati. Si noti che *i nomi delle tabelle della base di dati, dei relativi campi e delle relazioni tra di esse sono, per il momento, ipotetici. Per averne una descrizione puntuale si deve fare riferimento al capitolo dell'analisi tecnica e alla descrizione della base di dati.*

[1] Questo vale specialmente per piccoli progetti (della durata di meno di un mese) e per progetti ancora più contenuti, con impegni relativamente semplici. A mano a mano che le dimensioni e la complessità del software aumentano, questi argomenti iniziano a scricchiolare.

[2] Frederick Phillips Brooks, Jr. (nato a Durham, North Carolina, il 19 Aprile 1931) è un famoso ingegnere del software, maggiormente conosciuto per la gestione dello sviluppo del sistema operativo IBM OS/360 di cui parla nel suo famoso libro "The Mythical Man-Month". Egli ha ricevuto il premio Touring Award nel 1999.

### 8.1.1 Concetti fondamentali

In questa sezione verranno spiegati i concetti fondamentali su cui si basa il sistema. Si consiglia di leggere attentamente quanto segue per facilitare la comprensione di quanto scritto nei paragrafi successivi.

#### La baseline

Si definisce *baseline* una snapshot del sistema aziendale in un determinato istante. Essa identifica, quindi, l'organizzazione dell'azienda intesa come descrizione dell'organigramma e dell'insieme dei processi azienzali, tenendo conto delle attività che li compongono e dei documenti prodotti e/o utilizzati. Uno degli obbiettivi che il sistema deve conseguire è mantenere lo storico delle revisioni dei processi. Un processo può cambiare revisione anche solo per la sostituzione di un template di un documento oppure, ad esempio, per la modifica di una delle attività che lo compongono. Si consideri il secondo caso: l'unica cosa che cambia è un singolo elemento del grafo che rappresenta il processo, ma i restanti elementi del disegno non vengono modificati. Nonostante la soluzione ottimale sia unire la nuova attività con quelle non modificate senza copiare nulla, il problema è stato affrontato con un'approccio differente, in quanto questa strategia non risulterebbe banale da un punto di vista implementativo. Il cambiamento di un qualsiasi elemento di un processo comporta la creazione di una nuova baseline, in quanto è stato ritenuto un tabù modificare una baseline attiva all'interno dell'azienda, per non generare incongruenze tra i dati e per avere sempre la possibilità di capire con che baseline si stava lavorando quando un certo dato è stato prodotto nel passato. La differenza rispetto alla soluzione ottimale citata pocanzi consiste nel fatto che, nonostante alcuni elementi non vengano modificati, ne viene creata una nuova copia che sarà identificata dalla nuova baseline. Questo modo di operare garantisce, quindi, l'integrità delle precedenti revisioni dei processi e, quindi, la rintracciabilità dei documenti da essi prodotti come prescritto dalle normative.

#### Baseline management

Prima di iniziare a definire le possibili operazioni su una baseline, è necessario effettuare alcune considerazioni.
In un determinato istante, il sistema può gestire una singola baseline attiva ed una sola baseline in sviluppo. Sulla base di queste decisioni è stato disegnato il diagramma di stato del ciclo di vita di una baseline, rappresentato in figura 8.2. Lo stato iniziale *void* rappresenta la situazione in cui la baseline non è ancora stata creata. Al momento della sua crazione, in cui viene definito il nome, la baseline passa allo stato *development*, cioè in fase di disegno. Una baseline in sviluppo può essere cancellata mediante la funzione *drop*. Una volta concluso il lavoro di sviluppo, attraverso l'operazione *commit*, la baseline passa allo stato

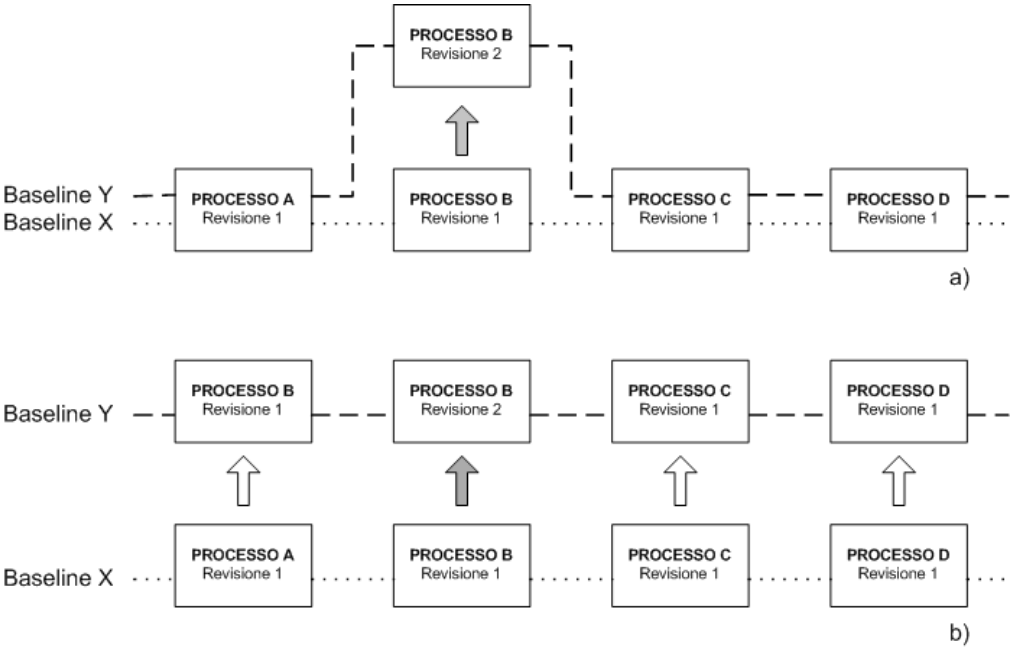


**Figura 8.1.** Rappresentazione grafica della progressione logica (a) e strutturale (b) dei processi.

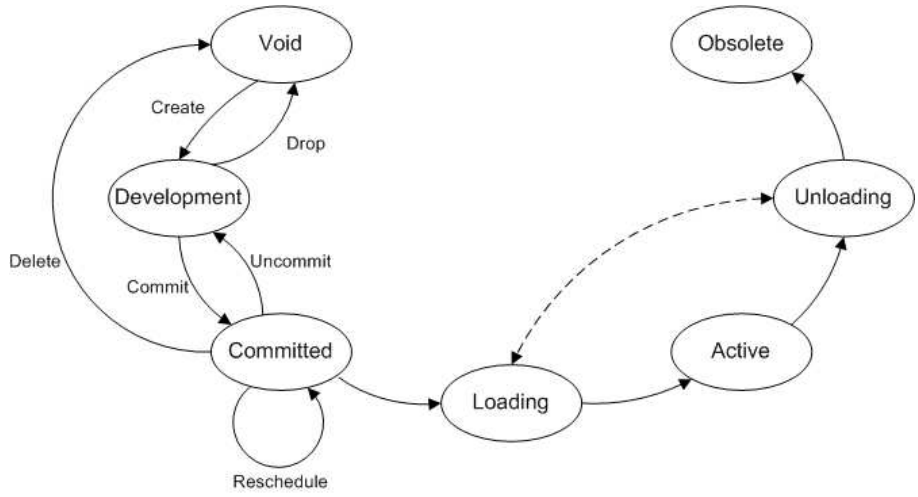


**Figura 8.2.** State diagram del ciclo di vita di una baseline.

*committed*. In pratica è stata definita la data in cui dovrà entrare in vigore e viene, quindi, gestita da un processo schedulatore che si occuperà della sua attivazione a tempo. Lo stato di una baseline committed può evolvere nei seguenti

modi:

- È possibile, tramite l'operazione *uncommit*, riportarla in sviluppo per apportare le dovute correzioni. Questa operazione viene effettuata se, ad esempio, ci si accorge di aver commesso alcuni errori in fase di disegno.

- Con l'operazione *delete* è possibile eliminare una baseline committed (la si deve selezionare da un elenco). In questo caso si ritorna allo stato iniziale.

- L'operazione *reschedule* consente di modificare la data di attivazione.

- Alla data di attivazione, lo schedulatore porta la baseline allo stato *loading*. Nel disegno è presente un collegamento tratteggiato tra lo stato loading e *unloading*. Questo indica semplicemente che il passaggio di una baseline allo stato loading comporta l'unload della baseline attiva. I due stati identificano una transazione, ovvero un insieme di operazioni che vanno eseguite in modo atomico. Per la durata della transazione non è consentito lavorare sul sistema. Terminata la fase di loading la baseline passa allo stato *active*: è, quindi, possibile generare istanze di processi in essa definiti. In seguito ad un'operazione di unload, una baseline passa allo stato *obsolete*: è solamente possibile concludere le attività relative ad istanze di processo precedentemente avviate.

### Subprocess management

Quando si vuole rappresentare graficamente un processo per studiarlo o gestirlo, è possibile farlo con diversi livelli di dettaglio. La rappresentazione, infatti, non è unica, perchè sono diversi i dettagli necessari, ad esempio, per la direzione, che vede il processo nelle sue linee essenziali, rispetto a quelli necessari per un tecnico, che deve, invece, trovare le informazioni per svolgere il suo compito. Esistono, dunque, diversi livelli di approfondimento e, quindi, di rappresentazione. Il disegno in figura 8.3 mostra diversi livelli di rappresentazione per uno stesso processo di esempio. Al primo livello si fornisce una raffigurazione globale e molto sintetica, scendendo sempre più nel dettaglio con i livelli successivi. Il grado di approfondimento è legato all'importanza e alla complessità del processo considerato.

Si illustra sinteticamente come il sistema di workflow andrà a gestire i sottoprocessi. Per maggiore chiarezza si ipotizzerà che lo stato di esecuzione dei processi assieme alla loro definizione sia descritto in una base di dati. Nel seguito si farà riferimento a una porzione di una possibile architettura, l'effettiva implementazione è tuttavia descritta nel capitolo riferito all'analisi tecnica. Nella rappresentazione grafica di un processo (non importa a che livello ci si trova), i sottoprocessi sono normalmente collegati alle attività "semplici" e vengono differenziati unicamente per il simbolo con cui sono rappresentati. A livello logico, invece, la differenza viene indicata in maniera esplicita dal contenuto del campo

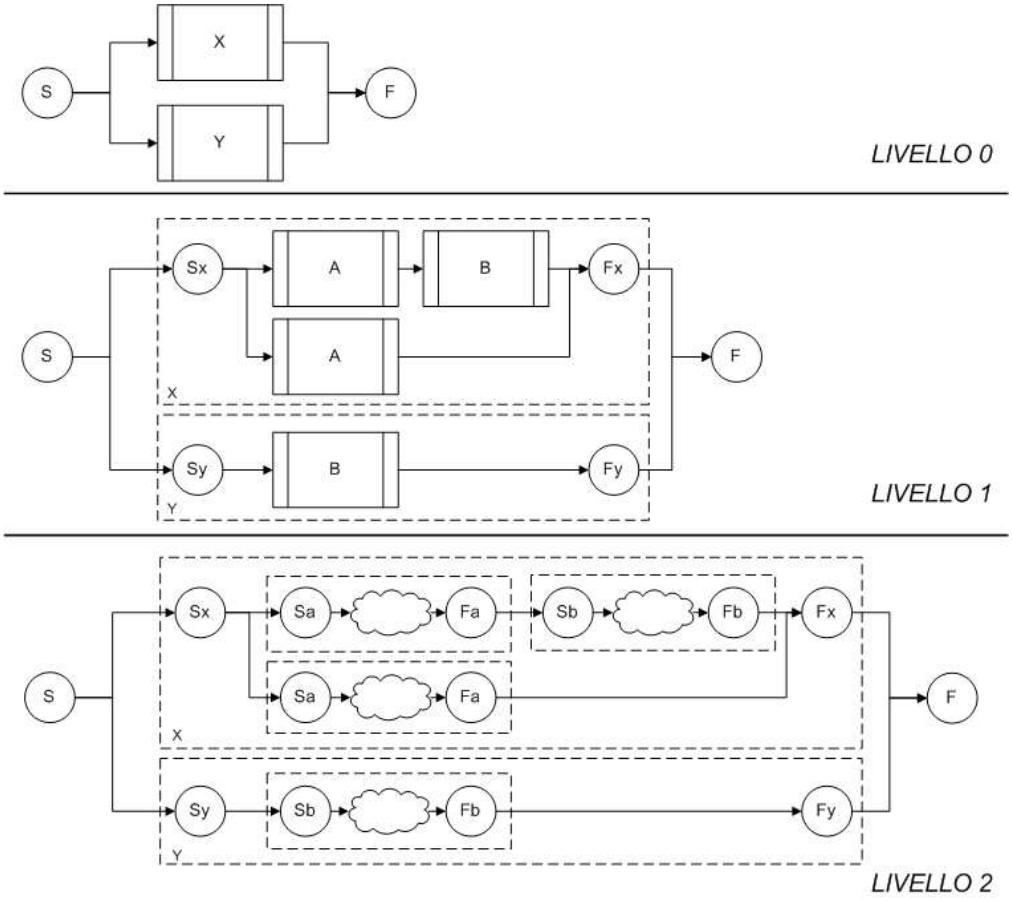


**Figura 8.3.** Diversi livelli di rappresentazione per un processo di esempio. Al livello 1, nel sottoprocesso X sono presenti due attività di tipo `subprocess` contrassegnate dalla lettera A. Esse sono due attività del tutto distinte, ma fanno riferimento allo stesso sottoprocesso. L'esecuzione delle due attività comporterà, quindi, l'apertura di due differenti istanze del processo A.

`type` nella tabella `Activity` che conterrà la voce `subprocess` Quando il sistema, percorrendo il grafo del processo, incontra un'attività che identifica un sottoprocesso, fa partire il processo associato identificato dal campo `lower_process_id`. Quando un processo viene terminato, il sistema controlla a che livello ci si trova attraverso il campo `level` della tabella `Process_trace`. Si possono verificare le seguenti situazioni:

- se il livello corrente è zero (`level`=0) ci si trova al livello superiore e, quindi, quello che è terminato è un macroprocesso;
- se il livello corrente è maggiore di zero (`level`$\geq$1), invece, significa che quello che è terminato è un sottoprocesso e, quindi, viene controllato il

valore del campo `father_id` associato, viene chiuso il processo e si prosegue l'esecuzione del processo padre al livello di dettaglio superiore.

Per chiarire meglio i concetti appena descritti, vediamo come le tabelle `Process`, `Activity`, `Process_trace` e `Activity_trace` potrebbero essere popolate con i processi/sottoprocessi dell'esempio di figura 8.3. I dati raccolti nelle tabelle descrivono lo stato di esecuzione rappresentato in figura 8.4.

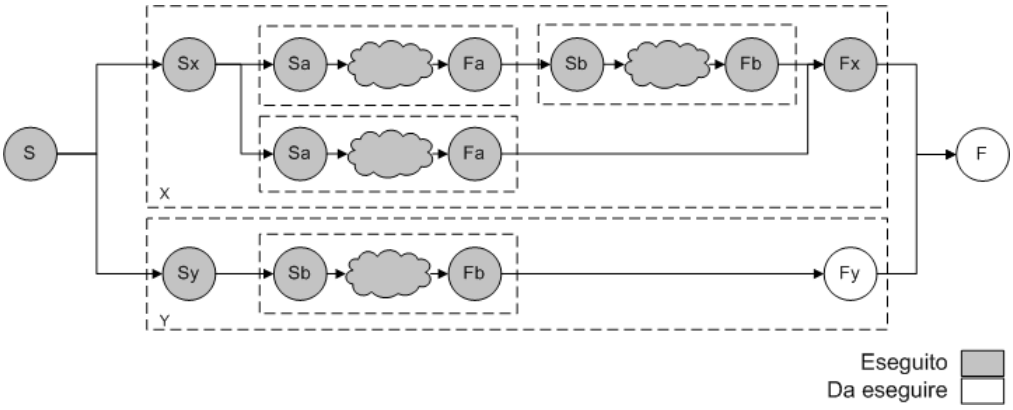


**Figura 8.4.** Stato di esecuzione del processo di esempio.

| id | name |
|---|---|
| 100 | X |
| 101 | Y |
| 102 | A |
| 103 | B |
| 104 | Macroprocesso |

**Tabella 8.1.** Estratto della tabella `Process`. Essa contiene i dati relativi ai processi aziendali e viene popolata in fase di disegno dei processi stessi.

| id | name | type | process_id |
|---|---|---|---|
| 201 | Sa | start | 102 |
| 202 | Fa | finish | 102 |
| 203 | Sb | start | 103 |
| 204 | Fb | finish | 103 |
| 205 | Sx | start | 100 |
| 206 | Fx | finish | 100 |
| 207 | Sy | start | 101 |
| 208 | Fy | finish | 101 |
| 209 | S | start | 104 |
| 210 | F | finish | 104 |
| 211 | X | subprocess | 104 |
| 212 | Y | subprocess | 104 |
| 213 | A1 | subprocess | 100 |
| 214 | B | subprocess | 100 |
| 215 | A2 | subprocess | 100 |
| 216 | B | subprocess | 101 |

**Tabella 8.2.** Estratto della tabella `Activity`. Questa entità raccoglie tutte le informazioni riguardanti le attività che compongono i processi. Anche queste informazioni sono definite in fase di disegno dei processi.

| id | instance | father_id | level | process_id |
|---|---|---|---|---|
| 001 | 34(Macroprocesso) | null | 0 | 104 |
| 002 | 46(X) | 34 | 1 | 100 |
| 003 | 47(Y) | 34 | 1 | 101 |
| 004 | 48(A) | 46 | 2 | 102 |
| 005 | 49(A) | 46 | 2 | 102 |
| 006 | 51(B) | 47 | 2 | 103 |
| 007 | 48(A) | 46 | 2 | 102 |
| 008 | 50(B) | 46 | 2 | 103 |
| 009 | 49(A) | 46 | 2 | 102 |
| 010 | 50(B) | 46 | 2 | 103 |
| 011 | 46(X) | 34 | 1 | 100 |

**Tabella 8.3.** Estratto della tabella `Process_trace`. Al contrario delle tabelle precedenti, questa viene popolata dal sistema durante l'esecuzione dei processi. Contiene i dati riguardanti le istanze di processo aperte e chiuse.

| id | instance | activity_id |
|---|---|---|
| 400 | 34(Macroprocesso) | 209 |
| 401 | 46(X) | 205 |
| 402 | 47(Y) | 207 |
| 403 | 48(A) | 201 |
| 404 | 49(A) | 201 |
| ... | 51(B) | 203 |
| ... | ... | ... |
| ... | 48(A) | 202 |
| ... | 50(B) | 203 |
| ... | ... | ... |
| ... | 49(A) | 202 |
| ... | 50(B) | 204 |
| ... | 46(X) | 206 |

**Tabella 8.4.** Estratto della tabella `Activity_trace`. Tiene traccia dello stato di avanzamento raccogliendo le informazioni riguardanti l'apertura e la chiusura delle diverse attività. Anch'essa viene popolata dal sistema run-time.

### Change management

Si assume che non ci possano mai essere più istanze attive di una stessa attività all'interno della stessa istanza di processo. Se un'attività già aperta subisce un tentativo di riapertura (ad esempio a causa di un cammino parallelo nel flusso di processo che riporta nell'attività corrente), il record nella tabella `Activity_trace` viene chiuso come se l'attività fosse stata conclusa, memorizzando anche che l'istanza corrente di attività è stata chiusa dal sistema per questo motivo. A questo punto viene immediatamente aperto un nuovo record corrispondente all'istanza di attività e in accordo al motivo per il quale l'attività è stata (ri)aperta. In questo record viene aggiornato, incrementandolo, il contatore del numero di riaperture dell'attività. Se la nuova riapertura è stata assegnatata come ACL (il significato di ACL verrà approfondito più avanti) a ruoli o login diversi da quelli dell'istanza precedente forzatamente chiusa, si rispettano, comunque, i requisiti dei nuovi valori di ACL. In altri termini, il lavoro assegnato all'istanza precedente viene forzatamente tolto a chi ci lavorava prima.

Nei record della tabella `Activity_trace` devono essere presenti due campi:

1. `change_count`: vale `0` quando si crea la prima istanza e tale resta se l'istanza viene chiusa naturalmente. Il nuovo record aperto a fronte di un "change" eredita il valore del contatore incrementato di un'unità.

2. `change_activity_trace_id`: vale `null` la prima volta. Viene impostato con il valore dell'`id` del nuovo record corrispondente alla nuova istanza di attività a fronte di un "change".

Quindi, a fronte di un "change", viene creato un nuovo record di istanza il quale:

- eredita tutti i riferimenti a documenti del record corrente che deve essere chiuso;
- contiene proprietà di ACL come se fosse stata aperta la prima volta (come verrà poi approfondito parlando di attività nei cicli);
- contiene lo stesso valore di `change_count` del vecchio record che sta per essere chiuso, incrementato di `1`;
- il campo `change_activity_trace_id` è impostato a `null`.

Il vecchio record viene chiuso impostando il campo `change_activity_trace_id` al valore di `id` del nuovo record. La pagina `JSP` che rappresenta l'attività in corso legge il valore di `change_count` e, se `>0`, segnala che si tratta di un rifacimento di attività e riporta il valore del contatore come numero di revisione dell'attività (`1`=revisione `1`).

La figura 8.5 mostra graficamente quanto descritto.

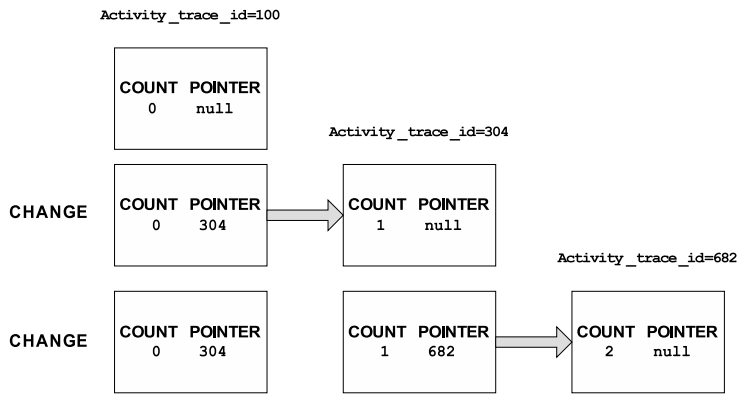


**Figura 8.5.** Esempio di change management. Gli attributi COUNT e POINTER corrispondono a `change_count` e `change_activity_trace_id`.

### Assignments

Per effettuare l'assegnamento delle risorse alle attività (con risorse intendiamo login) sono previste delle particolari attività che devono essere definite in fase di progetto del processo dal designer. Per meglio comprendere quanto detto, in figura 8.6 è riportato un semplice processo di esempio. Durante la definizione delle attività di assegnazione, il designer definisce l'insieme delle attività che saranno da esse gestite. In fase di esecuzione del processo, l'attività di assegnazione viene visualizzata dal suo owner (si tratterà di un responsabile, con

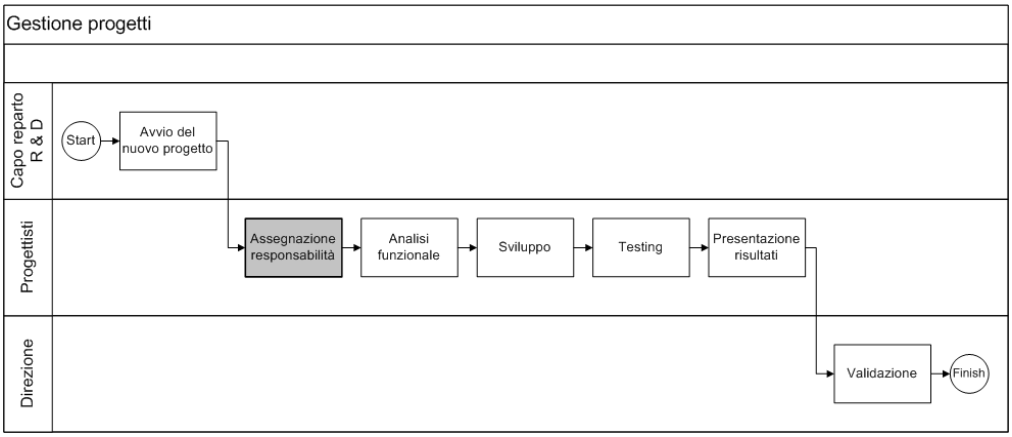


**Figura 8.6.** Processo di esempio. La figura evidenzia in grigio l'attività in cui avviene l'assegnazione delle risorse.

un ruolo che, quindi, avrà una corrispondenza uno a uno con una login). Nella vista relativa all'attività di assegnazione sarà presente la lista delle attività da assegnare e quella delle login relative ai ruoli definiti nella ACL per tali attività. Il responsabile non deve fare altro che mappare login su attività.

Ritornando al processo di esempio, l'attività che comporta l'assegnazione delle responsabilità sarà vista dal responsabile dei progettisti (ad esempio il capo progetto), il quale dovrà assegnare, per ipotesi, uno o più progettisti del suo team alle attività di analisi funzionale, sviluppo, testing e presentazione dei risultati.

### 8.1.2 System logon

All'avvio del sistema da parte di un qualsiasi utente, viene visualizzata una pagina in cui si chiede di inserire le credenziali per l'accesso. L'utente deve, quindi, digitare username e password ad esso assegnati. A questo punto, se i dati inseriti sono corretti, viene aperta la pagina principale del sistema, dalla quale ogni utente può accedere alle diverse funzionalità in base ai permessi assegnati (per esempio la funzione Baseline development, descritta nei paragrafi seguenti, sarà accessibile solamente a chi è stato assegnato il ruolo di process designer). Le funzionalità disponibili dipendono dall'appartenenza dell'utente (login) ad un gruppo. Gruppi distinti hanno privilegi distinti rispetto alle diverse funzionalità. Possono essere identificati i seguenti gruppi di utenti:

- process designer;
- system administrator;
- user;

- guest;
- observer.

I primi tre gruppi sono quelli che definiscono i compiti fondamentali: il disegno dei processi, la manutenzione del sistema e l'esecuzione quotidiana delle attività all'interno dell'azienda. Un utente appartenente al gruppo *guest* potrà visualizzare solo determinate informazioni, mentre un utente *observer* (ad esempio l'ispettore di un ente di certificazione) potrà vedere tutto, ma senza avere permessi di scrittura/esecuzione. La tabella 8.5 mostra sulle righe le funzionalità del sistema e sulle colonne i gruppi di utenti precedentemente individuati. Essa indica in che modo ogni gruppo può accedere alle diverse funzionalità.

| **Funzione** | **Permessi** | **Gruppi** | | | | |
|---|---|---|---|---|---|---|
| | | Process designer | System administrator | User | Guest | Observer |
| Baseline development | Read | Y | Y | N | N | Y |
| | Write | Y | Y | N | N | N |
| Process execution | Read | Y | Y | P | P | Y |
| | Write | N | Y | P | N | N |
| User administration | Read | Y | Y | P | N | Y |
| | Write | Y | Y | P | N | N |
| Data administration | Read | N | Y | N | N | Y |
| | Write | N | Y | N | N | N |
| Execute analysis | Read | P | Y | P | P | Y |
| | Write | P | Y | P | P | Y |

**Tabella 8.5.** Tabella dei permessi. Indica in che modo ogni gruppo di utenti può accedere alle diverse funzionalità del sistema. Legenda: Y=Yes, N=No, P=Partially. Quando è indicato un accesso parziale, significa che l'utente di quel gruppo potrà leggere/scrivere solamente le informazioni relative al suo ruolo.

### 8.1.3 Baseline development

Si ricorda che una baseline definisce staticamente i processi. Eventuali assegnazioni statiche di responsabilità possono, invece, essere definite sia contestualmente ai processi, sia durante l'esecuzione dei processi. La fase di edizione di una baseline segue i passi descritti in figura 8.7. Quando il sistema viene attivato per la prima volta non è presente alcuna baseline o processo e lo stato del sistema è detto *void*. Quando viene creata una baseline, il sistema passa allo stato *baseline development*. Le operazioni che si possono effettuare in questa modalità, come illustrato in figura 8.8, sono:

- create process;

- select and modify process;
- select and delete process;
- copy process from a baseline;
- select and verify process;
- view selected process;
- baseline commit;
- drop baseline;
- select and delete a committed baseline;
- reschedule a baseline.

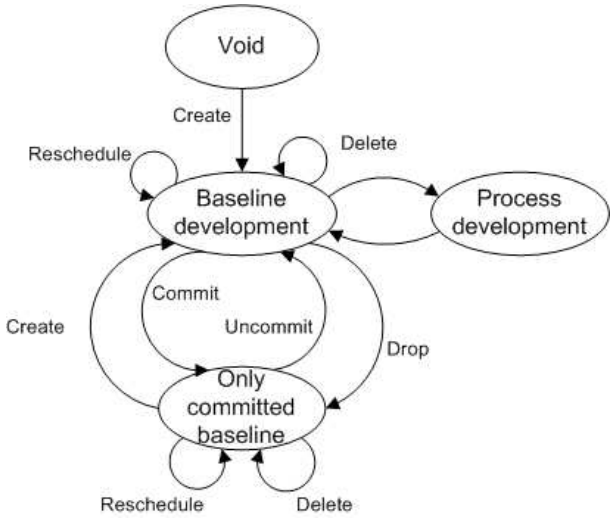


**Figura 8.7.** Diagramma di stato delle modalità operative per la definizione e attivazione di una baseline.

Quando viene effettuata un'operazione di *commit*, il sistema esce dallo stato *baseline development* e passa a *only committed baseline*. Come suggerito dal nome, in questa modalità non è possibile cambiare la definizione di una baseline, ma si può solamente cambiarne la schedulazione oppure effettuare la cancellazione di una baseline. È possibile tornare allo stato *baseline development* effettuando un'operazione di *uncommit* su una baseline schedulata ma non ancora attivata (una baselina *committed*). Un altro modo per tornare allo stato *baseline development* è creare una nuova baseline con l'operazione *create baseline*.

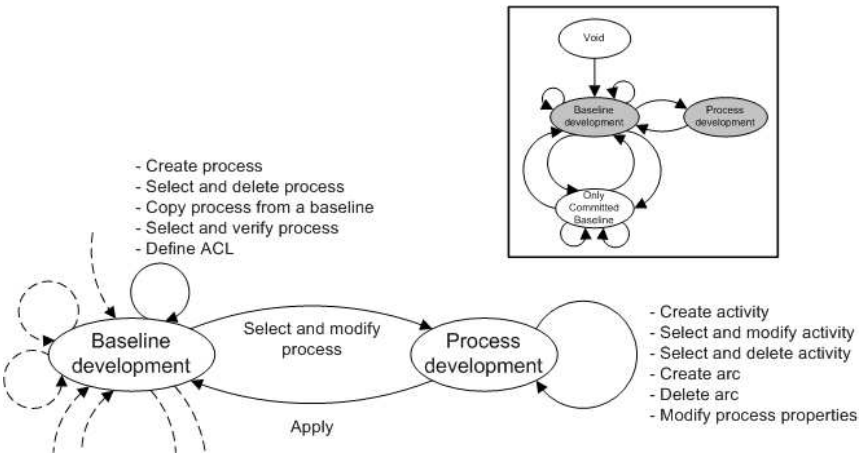


**Figura 8.8.** Dettaglio del diagramma di stato delle modalità operative per la definizione dei processi.

### Create process

Il sistema chiede di specificare in forma obbligatoria il nome del processo da creare. Le informazioni riguardanti la revisione del processo, la baseline di appartenenza e la data di attivazione sono, invece, gestite in automatico dal sistema nel momento in cui viene effettuata l'operazione di *commit*. Un processo così creato si presenta come una scatola vuota; per poterne modificare il "contenuto", il process designer utilizzerà la funzione *select and modify process*.

### Select and modify process

Attraverso la funzione *select and modify process* è possibile selezionare tra la lista dei processi precedentemente creati quello che si desidera modificare. Il sistema passa, quindi, alla modalità *process development* che verrà descritta nella sezione seguente.

### Select and delete process

Mediante la funzione *select and delete process*, dalla lista dei processi creati nella baseline in sviluppo è possibile selezionare il processo da eliminare.

### Copy process from a baseline

Attraverso la funzione *copy process from a baseline* è possibile aggiungere uno o più processi alla baseline in disegno copiandoli direttamente dalla baseline attiva o da quelle utilizzate in passato. Una tabella simile a quella di figura 8.9 indica, per ogni processo (righe), le baseline in cui è presente. Per aggiungere il

processo alla baseline in sviluppo e sufficiente selezionare le righe corrispondenti della prima colonna.

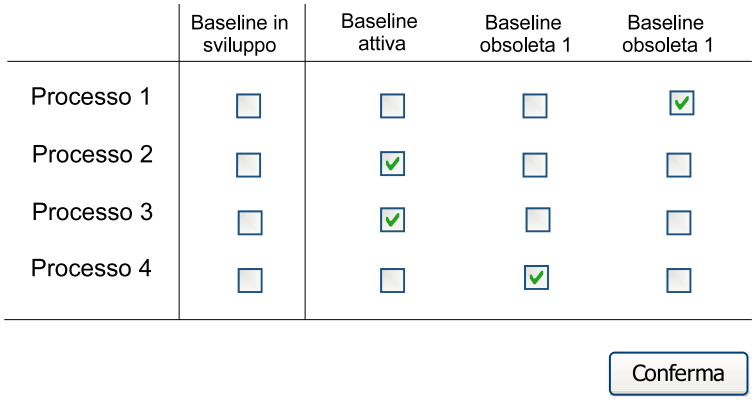


**Figura 8.9.** Esempio della fase di interazione con l'utente relativa alla copia di un vecchio processo.

### Select and verify process

Utilizzando la funzione *select and verify process* è possibile avviare una routine che verifica la consistenza del disegno del processo. La procedura effettua i seguenti controlli:

- Controlla che ogni attività del processo in fase di modifica, abbia almeno un arco entrante ed uno uscente. Per l'attività start è sufficiente averne uno in uscita, mentre, per quella conclusiva, almeno uno in entrata.
- Verifica che tutte le attività create siano raggiungibili dal nodo di partenza. Questo fase include, perciò, la verifica che il nodo finish sia raggiungibile dal nodo start tramite almeno un cammino..
- Verifica che il processo analizzato non possa mai essere avviato, ovvero che non abbia un owner per l'attività start e, contemporaneamente, non esista un processo che lo invochi come sottoprocesso.
- Verifica che sia stato definito l'owner del processo.
- Controlla che la definizione dei documenti in input/output delle attività sia coerente. Se, per esempio, per un'attività A viene indicato di utilizzare in input il documento X prodotto dall'attività B, ma B non produce tale documento, il sistema segnalerà l'incongruenza.

Al termine dell'elaborazione della routine vengono riportate le attività che risultano "inconsistenti".

### View selected process

La funzione *view selected process* offre, a chi sta disegnando i processi, la possibilità di avere una visione di insieme del lavoro svolto. Sono disponibili le seguenti scelte:

- Data una baseline, visualizza tutti i processi relativi ad essa. Se si seleziona la baseline in disegno si ha, quindi, una visione dello stato dei lavori. Deve essere possibile scegliere il livello di astrazione della descrizione (decidere, per esempio, se mostrare o meno i sottoprocessi).

- Scelto un processo, visualizza le attività e gli archi ad esso relativi con i corrispondenti dettagli.

### Baseline commit

Con la funzione *baseline commit* il process designer schedula la baseline disegnata. L'entrata in vigore di una baseline blocca lo start di tutti i processi di quella precedente, ma permette, comunque, la conclusione di tutti quelli precedentemente avviati.

Il process designer deve indicare la data alla quale desidera rendere operativa la baseline. Prima di eseguire il vero e proprio *commit*, il sistema verifica la correttezza del disegno dei processi e, nel caso il test non vada a buon fine, mostra i motivi e non schedula la baseline. Se, invece, il test viene passato, il sistema memorizza la data in cui attivare la baseline e la invia ad un processo schedulatore che si occuperà della sua attivazione a tempo. Non è assolutamente possibile modificare una baseline attiva. È invece possibile far ritornare una baseline committed allo stato di progetto purchè non sia già entrata in vigore: in tal caso viene cancellata la richiesta allo schedulatore.

### Drop baseline

La funzione *drop baseline* consente di eliminare la baseline in sviluppo. Questo comporta il passaggio automatico del sistema alla modalità only committed baseline.

### Select and delete a committed baseline

La funzione *select and delete a committed baseline* consente di eliminare una qualsiasi baseline committed ma non ancora avviata. La funzione è attivabile sia dalla modalità *baseline development* che da quella *only committed baseline*.

### Reschedule a baseline

L'operazione *reschedule a baseline*, se eseguita in modalità *baseline development*, consente di modificare in qualsiasi momento la data di schedulazione della base-

line in sviluppo. Se si richiama la funzione in modalità *only committed baseline*, invece, è possibile cambiare la data di avvio dell'ultima baseline committed.

### 8.1.4 Process development

Quando si seleziona un processo da modificare tramite la funzione *select and modify process*, il sistema passa allo stato *process development*. Le operazioni che si possone effettuare in questa modalità (vedi figura 8.8) sono:

- create activity;
- select and modify activity;
- select and delete activity;
- create connector;
- delete connector;
- modify connector;
- define ACL;
- modify process properties.

#### Create activity

Per creare una nuova attività, l'utente deve indicare il nome, il tipo ed, opzionalmente, una descrizione dell'attività stessa. Il sistema memorizza permanentemente tali dati. I campi che si riferiscono al processo e alla baseline associata sono definiti dalla stato corrente del sistema. Come per i processi, anche le attività vengono create come delle scatole vuote. Per definirne il contenuto il process designer deve utilizzare la funzione *select and modify activity* a breve illustrata.

#### Select and modify activity

Mediante la funzione *select and modify activity* è possibile modificare una attività precedentemente creata selezionandola da una lista che elenca tutte le attività create per il processo in modifica. Con questa funzione è possibile modificare i campi compilati al momento della creazione dell'attività (vedi *create activity*): la pagina mostra un form identico nel quale, però, i campi sono precompilati con i dati precedentemente immessi. Per un'attività creata si devono, inoltre, indicare i documenti utilizzati (input) o prodotti (output). Deve essere, quindi, possibile sceglierli tra i documenti presenti nella base di dati. Il process designer deve inserire le informazioni utili al sistema per "disegnare" i form nelle pagine che verranno viste dagli utenti in fase di esecuzione delle attività. I documenti in output di un'attività saranno generalmente caricati nel form attraverso un

comando "Carica" che fa selezionare il documento dal file system locale. I documenti in input, invece, sono file che, al momento dell'esecuzione dell'attività, sono già stati creati. Si possono verificare due diverse situazioni:

- *Static reference*: il link al documento è memorizzato nella base di dati in fase di disegno. Un documento che rientra in questa categoria può essere, per esempio, un manuale di istruzioni di uno strumento o un template.

- *Cross reference*: il documento in input viene generato da un'altra attività in fase di runtime. Il link al documento sarà, quindi, disponibile solo alla conclusione dell'attività produttrice. `null`).

Quando viene generata la pagina che l'utente vede quando deve svolgere un'attività, il sistema deve fornire i link ai documenti in ingresso esplicitando il nome del documento che si deve consultare per svolgere il lavoro. Allo stesso modo il sistema deve indicare il nome dei documenti che si aspetta di ricevere in ingresso. Per poter fare tutto ciò, però, è necessario che queste informazioni vengano inserite nel sistema quando le attività vengono disegnate. Per spiegare con maggior chiarezza ciò che si è appena detto si consideri il seguente esempio: l'attività *implementation design* del processo *development* produce in output il documento *functional specifications*. In figura 8.10 è raffigurata un'ipotesi di come si potrà presentare all'utente il form per effettuare l'upload del documento. Nella stessa immagine sono, inoltre, evidenziate le informazioni che devono essere note al sistema perchè esso possa correttamente disegnare il form.

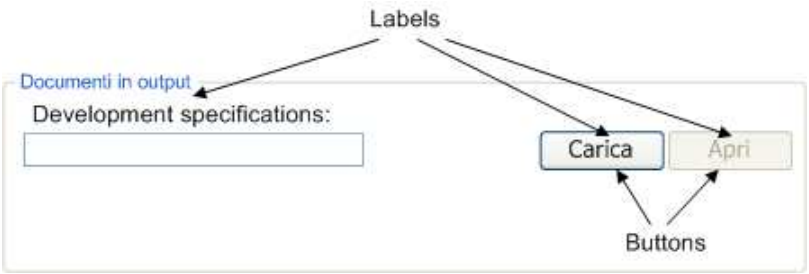


**Figura 8.10.** Informazioni per il rendering dei form.

### Select and delete activity

L'utente deve semplicemente selezionare l'attività da eliminare dalla lista di tutte le attività create per il processo in modifica e premere un pulsante di conferma. In automatico il sistema elimina gli archi ad essa associati, entranti ed uscenti, lasciando, in alcuni casi, disconnesso il grafo che rappresenta il processo. Controlli sul grafo del processo sono svolti in un secondo momento.

### Create connector

Visto che nel grafo di un processo gli archi sono orientati, si prevede di inserirli specificando l'attività da cui partono e l'attività in cui arrivano, scegliendo queste tra le attività già inserite per il processo in fase di modifica. Deve essere possibile inserire anche la *label* del connettore (d'ora in poi sarà utilizzato questo termine). Le label sono obbligatorie per tutti i connettori che escono da un'attività che prevede una scelta (per fare un esempio, dall'attività "approvazione della direzione" partiranno due archi con etichette "si" e "no"). Risulta utile, una volta selezionata l'attività da cui parte il connettore, poter visualizzare i connettori uscenti già presenti, per evitare di inserire connettori ridondanti.

### Delete connector

Tramite la funzione *delete connector* viene visualizzata la lista di tutti i connettori già inseriti per il processo in modifica in una tabella strutturata come quella raffigurata nella figura 8.11. È possibile eliminare più connettori con un'unica operazione.

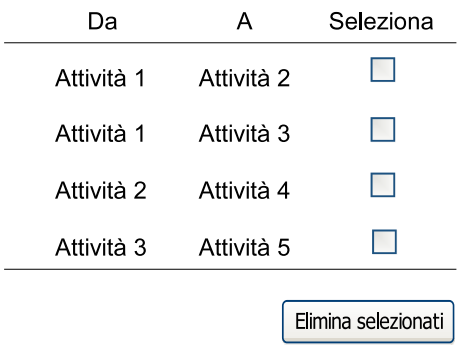


**Figura 8.11.** Lista dei connettori eliminabili.

### Modify connector

L'operazione di modifica di un connettore non è prevista. Tale scelta è stata fatta osservando la semplicità delle operazioni di creazione e cancellazione. Se ci si accorge di aver commesso un errore nella definizione di un connettore, quindi, non si deve far altro che eliminarlo e crearne uno nuovo con i dati corretti. Si prevede, comunque, fin da subito una non specifica di funzione (ovvero *modify connector* per eventuali release successive.

### Define ACL

Attraverso la matrice di responsabilità è possibile definire i permessi per le attività del processo selezionato, ovvero a quali ruoli sono visibili e con che modalità. L'utente avrà a disposizione una tabella simile a quella rappresentata in figura 8.12. Oltre a definire i permessi per le attività, è possibile esplicitare anche

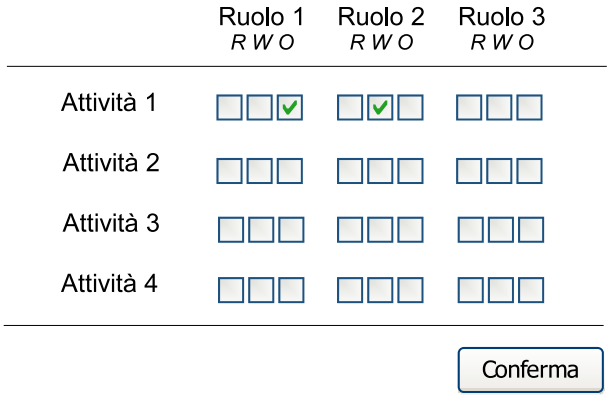


**Figura 8.12.** Modifica dei permessi.

l'owner del processo ed i meccanismi di escalation. Quest'ultimi specificano in che modo il sistema si deve comportare qualora entri in esecuzione un'attività senza risorse assegnate. È possibile utilizzare le seguenti *policy*:

- *First In First Work (FIFW) con conferma di aquisizione responsabilità*: con questa politica, la prima persona che apre l'attività (deve appartenere ad uno dei ruoli definiti con permesso Owner) si prende carico del lavoro sulla stessa. Al momento dell'apertura dell'attività, viene visualizzato un messaggio con una richiesta di conferma. In pratica si dà all'utente la possibilità di rifiutare l'assegnazione automatica. É, inoltre, possibile specificare opzionalmente se l'acquisizione di attività esclude automaticamente tutti gli utenti del gruppo oppure no. In altri termini, se si opta per l'esclusione, quando un utente si prende carico di un'attività, questa scompare dalla lista delle attività da eseguire degli altri utenti con lo stesso ruolo.

  Vengono, ora, presentati alcuni esempi per chiarire il funzionamento delle politiche FIFW. In caso di FIFW non esclusiva (esempio nella tabella 8.6), quando un utente conferma l'accettazione della responsabilità (diventa owner), gli utenti dei ruoli che avevano permesso 'O' abilitato lo perdono. In questo caso anche gli utenti con ruolo "jolly" (Matteo) perdono la possibilità di diventare owner. In caso di FIFW esclusiva (vedi esempio nella tabella 8.7), invece, quando un utente conferma l'accettazione della responsabilità (diventa owner), gli utenti dei ruoli che avevano permesso

| Login | Ruolo | R | W | O | | Login | Ruolo | R | W | O |
|---|---|---|---|---|---|---|---|---|---|---|
| Andrea | comm. | X | | X | | Andrea | comm. | X | | X |
| Enrico | comm. | X | | X | *Andrea* | Enrico | comm. | X | | |
| Mauro | dir. | X | | | *accetta* | Mauro | dir. | X | | |
| Enrico | prog. | | X | | —> | Enrico | prog. | | X | |
| Matteo | jolly | X | X | X | | Matteo | jolly | X | X | |

**Tabella 8.6.** Azioni sui permessi in seguito all'accettazione di un'attività FIFW non esclusiva.

'O' abilitato lo perdono. Inoltre, per gli utenti di quei ruoli, vengono inibiti anche tutti gli altri permessi.

| Login | Ruolo | R | W | O | | Login | Ruolo | R | W | O |
|---|---|---|---|---|---|---|---|---|---|---|
| Andrea | comm. | X | | X | | Andrea | comm. | X | | X |
| Enrico | comm. | X | | X | *Andrea* | Enrico | comm. | | | |
| Mauro | dir. | X | | | *accetta* | Mauro | dir. | X | | |
| Enrico | prog. | | X | | —> | Enrico | prog. | | | |
| Matteo | jolly | X | X | X | | Matteo | jolly | | | |

**Tabella 8.7.** Azioni sui permessi in seguito all'accettazione di un'attività FIFW esclusiva.

- *Escalation all'owner del processo*: in questo caso il sistema invia una notifica all'owner del processo. Sarà suo compito assegnare la risorsa.
- *Escalation al responsabile della banda funzionale (Group Activity Owner)*: simile al caso precedente, questa policy indica al sistema di inviare una notifica non all'owner del processo, ma al ruolo superiore a quello assegnato all'attività nell'ACL (i ruoli mappati in `boss_id` della tabella `role` si riferiscono, generalmente, ad una singola persona).

La presenza di cicli nei processi comporta alcuni casi particolari nell'applicazione delle politiche di assegnamento:

- *FIFW in un ciclo*: quando si arriva ad eseguire un'attività con politica di assegnamento FIFW si deve sempre controllare che non siano già state create istanze della stessa attività nella medesima istanza di processo. Questo significa verificare la presenza o meno di un ciclo. Se l'attività viene eseguita per la prima volta essa viene resa disponibile a tutti gli utenti che hanno la facoltà di impossessarsene. Se, invece, l'attività rientra in un ciclo (viene rieseguita), esse deve essere assegnata a colui che l'aveva svolta l'ultima volta. Solo una modifica dell'assegnazione in runtime può modificare tale situazione.

- *Owner escalation in un ciclo*: la politica owner escalation è quella di default e viene applicata quando si vuole che l'assegnazione venga fatta in base a decisioni prese all'istante. Per questo motivo si ritiene opportuno lasciare che la decisione venga presa di volta in volta, anche in presenza di cicli (una persona prima disponibile ora può non esserlo più).
- *GAO in un ciclo*: un'attività di tipo GAO all'interno di un ciclo viene essenzialmente ripetuta e i suoi assegnamenti sovrascrivono quelli precedenti. Un'attività con politica GAO, invece, nel caso venga ripetuta perchè contenuta in un ciclo, eredita semplicemente l'assegnamento della sua istanza precedente come per le politiche FIFW. Il fatto che, al momento della sua apertura, un'attività con politica di assegnamento GAO risulti non assegnata fa immediatamente capire che essa fa parte di un ciclo.

#### Modify process properties

Utilizzando questa funzione è possibile modificare tutte le proprietà definite al momento della creazione del processo selezionato. Inoltre deve essere offerta all'owner di processo la possibilità di abortire un'istanza di processo in qualsiasi momento.

### 8.1.5 Process execution

Le operazioni per l'esecuzione dei processi sono quelle che verranno eseguite dall'utente che, quotidianamente, andrà ad utilizzare il sistema. Sono state individuate le seguenti funzioni:

- start process;
- view active process instances;
- execute activity.

#### Start process

Tramite la funzione *start process* l'utente visualizza l'elenco di tutti i processi da lui attivabili. Questo significa vedere tutti i processi per i quali si possiede il permesso *owner* per l'attività di start. L'owner dell'attività start, ovvero colui che può creare una nuova istanza di un processo, non coincide necessariamente con l'owner del processo. Si pensi, ad esempio, al processo di gestione dei guasti di una compagnia che offre un servizio di telefonia fissa. Il cliente che riscontra un malfunzionamento della rete effettua una chiamata al numero dell'assistenza clienti. L'operatore dell call center che riceve la telefonata, dopo aver raccolto le informazioni, apre la pratica del guasto, ma non sarà certo lui il responsabile della pratica. A questo punto è necessario definire il nome dell'istanza di processo (ad esempio, per il processo "Offerta commerciale", il nome di un'istanza può essere "Offerta Rossi").

#### View active process istances

L'utente visualizza la lista delle attività da svolgere ordinate secondo un criterio selezionabile (ordine alfabetico, codice istanza di riferimento, data di attivazione processo, ...). Ogni riga dell'elenco mostra il nome dell'attività ed i dati relativi all'istanza di processo al quale fa riferimento, quali nome istanza, nome processo, owner, data inizio, scadenza[3]. Selezionando l'attività desiderata, in un frame nella stessa pagina, compaiono tutte le informazioni utili al suo svolgimento (vedi Esecuzione attività).

#### Execute activity

Una volta scelta l'attività da svolgere, l'utente visualizza i dettagli della stessa utili allo svolgimento. Vengono, quindi, visualizzati i link ai documenti in input, i form da compilare per effettuare l'upload dei documenti in output, una descrizione dell'attività ed i comandi per proseguire all'attività successiva. Ovviamente un utente può interagire con la pagina in funzione dei permessi associati al ruolo posseduto. Il pulsante *Next*, per esempio, sarà abilitato solamente per gli utenti con permesso *owner*, i documenti potranno essere caricati solo da chi ha il permesso di scrittura, mentre chi ha solo i permessi di lettura può solemente essere informato dello stato di avanzamento dell'attività.

### 8.1.6 Users administration

A questa sezione possono accedere solamente coloro che godono dei diritti di amministrazione utenti/ruoli. La pagina presenta un pannello comandi dal quale è possibile selezionare le seguenti funzioni:

- create login:
- delete login;
- modify login;
- create role;
- delete role;
- modify role;
- assign role.

Essendo utili al disegno di un processo, le operazioni sui ruoli sono accessibili anche dal process designer.

[3] il campo *scadenza* della tabella non è stato definito nel progetto iniziale della base di dati, ma questa sembra un'informazione da tenere in considerazione. Si rimanda al capitolo dell'analisi tecnica

#### Create login

L'amministratore crea una nuova login indicando lo username. La password viene generata in automatico dal sistema.

#### Delete login

Nella pagina compare una lista dalla quale è possibile selezionare le login da eliminare.

#### Modify login

I dettagli di ciascuna login (username e password) sono visibili in una lista. L'amministratore può modificare entrambi i campi. Questa funzione è disponibile in forma *ligth* ad ogni utente per permettergli di modificare la propria password.

#### Create role

Per creare un nuovo ruolo si deve inserirne il nome (campo obbligatorio) e selezionare assigner e boss. È, inoltre, possibile inserire una breve descrizione del ruolo.

#### Delete role

La funzione consiste in una semplice cancellazione di un ruolo selezionato da un elenco. Il sistema avvisa l'utente che la cancellazione di un ruolo può rendere "orfani" processi e/o attività.

#### Modify role

Viene caricato un form simile a quello visualizzato in fase di creazione. È possibile modificare i valori di tutti i campi che sono precompilati con i dati precedentemente inseriti.

#### Assign role

Selezionando un ruolo è possibile visualizzare una lista di tutte le login ad esso associate, con la possibilità di aggingerne/toglierne.

### 8.1.7 Data administration

Le operazioni dirette sui dati del database sono estremamente pericolose perchè possono facilmente portare ad inconsistenze. Per questo motivo si reputa fondamentale effettuarle solamente in casi limitati, ad esempio in presenza di gravi errori. Queste azioni devono essere svolte da un *superuser* qualificato che abbia

un'ottima conoscenza della struttura della base di dati e degli strumenti per interagire con essa (DBMS e linguaggio SQL). Le operazioni di manutenzione sul DB possono essere effettuate utilizzando i tools messi a disposizione dal DBMS.

### 8.1.8 Execute analysis

Le operazioni di analisi dei dati corrispondono alla generazione di report da parte degli utenti del sistema. In seguito ai colloqui con l'azienda è emerso che risulta difficile individuare a priori quali interrogazioni sulla base di dati andranno a comporre la reportistica, in quanto le esigenze di un'azienda in termini di analisi dei dati possono variare rapidamente nel tempo. Preso atto di tutto ciò, si è, comunque, concordata la necessità di fornire agli utenti una libreria di report già pronti che potrà essere, in seguito, estesa dagli utenti stessi. Alcuni possibili report da inserire inizialmente nella libreria sono di seguito elencati.

- *Elenco delle attività pendenti per ogni login.* I dati prodotti da questo report possono essere utilizzati, ad esempio, per valutare se assegnare o meno un'attività ad una persona. Nel caso una persona risulti oberata di lavoro è possibile provvedere ad una riassegnazione delle attività. Deve essere possibile assegnare le attività per login, ruolo o processo.

- *Elenco di tutte le attivitá aperte da più di n giorni.* Il report mostra le attività che risultano "bloccate" permettendo alla direzione di individuare i colli di bottiglia nei diversi processi aziendali.

- *Elenco dei documenti prodotti da un determinato processo in un dato intervallo temporale.*

- *Elenco dei documenti prodotti da un determinato processo a seconda della revisione.*

- *Elenco delle revisioni di un documento di una determinata istanza processo.*

- *Elenco di tutti i documenti creati da un ruolo/persona.*

Oltre ai report di esempio appena riportati, deve essere possibile effettuare delle interrogazioni sulla base di dati che si differenziano da i report per il fatto che non forniscono una lista di elementi, ma un dato diretto. Alcuni esempi sono:

- *Ricerca della persona che ha prodotto un determinato documento.*

- *Durata media di un determinato processo* (intesa come media della durata delle sue istanze).

- *Durata di un'attività.*

- *Numero di riesecuzioni di un'attività in un ciclo.*

Riassumendo, quindi, il sistema deve dare la possibilità all'utente di generare report personalizzati in qualsiasi momento. Realizzare un'interfaccia grafica integrata al sistema per la creazione e la gestione della reportistica non risulta semplicissimo. Per questo motivo, vista la presenza sul mercato di numerosi *report generators*, si è pensato di ricorrere ad un prodotto già esistente da interfacciare con l'architettura del sistema. Nel capitolo dell'analisi tecnica verranno approfonditi i requisiti del report generator e verrà effettuata la scelta di quello da utilizzare nel progetto WQF.

# Capitolo 9

# Analisi tecnica

## 9.1 Architettura dell'applicazione

La progettazione di un'applicazione richiede sempre di affrontarne lo studio dell'architettura complessiva. Chiarire tale tema consente una maggior coordinazione del codice con un complessivo vantaggio nella comprensione e nella leggibilità dell'applicazione stessa. Inoltre, ciò consente l'uso di un linguaggio unificato tra gli sviluppatori che permette loro una maggiore comprensione degli intenti reciproci. Infine, l'architettura scelta impone un orientamento alle successive implementazioni e sviluppi e garantisce un'omogeneità sempre desiderabile quando un lavoro viene svolto in gruppo.

### 9.1.1 I livelli di astrazione

Quando si costruisce un'applicazione, nella fattispecie una *web application*, è bene distinguere diversi livelli di astrazione (in inglese *layer*). Si tratta di astrazioni logiche che interagiscono una con l'altra, ma che sono isolate al tempo stesso. Un layer può essere, ad esempio, la struttura che si occupa di organizzare la persistenza dei dati (*persistence layer*) che avrà a che fare solamente con il layer che si occupa delle operazioni di servizio (*service layer*). Questi due layer sono collegati tra loro, ma il service layer isola il persistence layer dagli altri. Non sempre però si verifica tale isolamento, come nel caso del *domain model layer*. Ciò sarà più chiaro nel proseguio di questo capitolo. Tipicamente un'applicazione web viene suddivisa in tre livelli:

- il *Top layer* che si occupa delle interazioni con l'utente;
- il *Middle layer* che fornisce i servizi di base;
- il *Bottom layer* che si occupa della persistenza dei dati;

ciascuno dei quali ha diretto rapporto con il solo suo superiore in una struttura come quella illustrata in fig. 9.1.

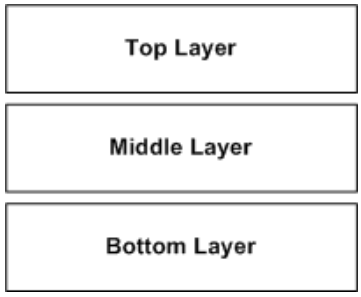


**Figura 9.1.** Tradizionale suddivisione dei layer in un'applicazione web.

Con uno sguardo più approfondito si possono identificare cinque layer di astrazione:

- lo *User Interface layer*;
- il *Web layer*;
- il *Service layer*;
- il *Domain model layer*;
- il *Persistence layer*;

anch'essi collegati in una struttura dove *User Interface layer* e *Web layer* corrispondono al *Top layer*, il *Service layer* al *Middle layer* e il *Persistence layer* al *Bottom layer*. Il *Domain Model layer* può essere considerato trasversale agli altri quattro. La figura 9.2 illustra la situazione appena descritta.
Come appena detto i layer sono tra loro isolati, nel senso che ciascuno ha diretto

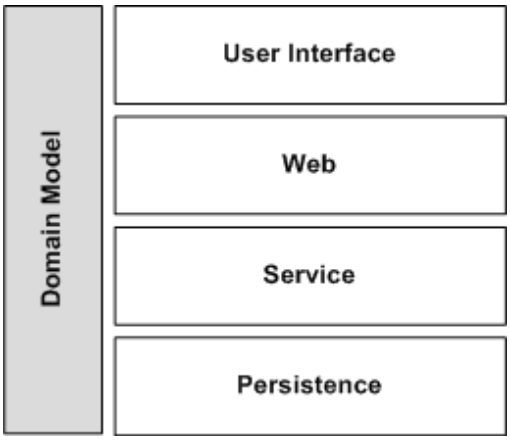


**Figura 9.2.** Suddivisione dettagliata dei layer in un'applicazione web.

rapporto solo con il suo immediato superiore. Ciò consente di separare il dominio dei problemi da affrontare (ad esempio la persistenza dei dati, la navigazione web o la realizzazione dell'interfaccia utente) e conduce verso un'applicazione più flessibile e testabile. Così facendo, infatti, l'implementazione di ciascun layer può variare in modo indipendente, con evidente beneficio per la flessibilità, favorendo, quindi, il disaccoppiamento delle relazione di dipendenza tra di essi (da qui la maggior testabilità). Favorire l'isolamento significa ridurre le dipendenze tra i layer: minori sono le dipendenze, meno risulta costosa la modifica di un layer. Per questo motivo si deve fare in modo che ogni layer comunichi solo con uno o due livelli, il Domain Model layer rappresenta, ovviamente, un'eccezione.
Per capire quanto detto detto finora, è necessario approfondire il significato dei cinque layer, le loro interazioni e funzioni specifiche. Si dedica, perciò, ancora del tempo a questo tema, anche per sottolinearne l'importanza.

### User Interface layer

Lo User Interface layer si occupa di presentare le risposte generate dal web layer all'utente. Tali informazioni si possono essere fornite in varie forme come, ad esempio, un file `PDF` o `XML`, oppure una pagina `XHTML` per un browser. In linea generale questo layer deve essere tenuto quanto più possibile separato dal web layer in modo da riutilizzare quest'ultimo quanto più possibile. Esso rappresenta l'ultimo livello raggiunto dall'applicazione una volta che sono state raccolte e organizzate tutte le informazioni necessarie. Tenere lo User Interface layer come ultimo livello e separato dal resto dell'applicazione consente di non vincolare, per esempio, le altre risorse ai tempi di connessione e trasmissione dei dati, lasciandole, quindi, libere per altri servizi. Inoltre, si può rendere più agevole la migrazione tra varie tecnologie di visualizzazione come `FreeMarker`, `JSP`[1], `Velocity`, `XSLT` o `Tiles`, poiché tale passaggio non influenza gli altri layer. Ciò rende anche più facile la collaborazione tra sviluppatori che si specializzano in aree diverse, nella fattispecie nel design di pagine web, consentendo a questi ultimi uno sviluppo parallelo con l'utilizzo degli strumenti ad essi più congeniali.

### Web layer

Il Web layer ha due funzionalità principali. Innanzitutto esso dirige il flusso di navigazione dell'utente in modo da rappresentare nella sequenza coretta la logica di navigazione tra le varie viste; in secondo luogo stabilisce il collegamento tra il Service layer e il mondo dell'`HTTP`. Il Web layer delega ogni logica di servizio al Service layer e si occupa di gestire i parametri di richiesta e di risposta, gestisce le sessioni `HTTP` e, in generale, interagisce con le `Servlet API`. Il Web layer chiama, quindi, i metodi del Service layer e gestisce gli errori da inviare all'utente, cosa che rende molto più semplice la navigazione allo stesso.

[1] http://java.sun.com/products/jsp/

### Service layer

Il Service layer occupa un ruolo importante sia per il client, ovvero chi ne invoca i metodi, sia per il sistema. Al primo fornisce funzionalità a grossa granularità per un utilizzo facile, non imponendogli di dover invocare una serie complessa e intricata di metodi e, al tempo stesso, limitandone al massimo le interazioni con il resto del sistema. Per grossa granularità si intende che il client, con una chiamata a un metodo dell'interfaccia di servizio, compie tutto il lavoro di cui ha bisogno senza doversi occupare di quali metodi ausiliari siano da invocare e quale sia il loro ordine corretto. Realizzare tutto il lavoro a livello del Service layer consente di ridurre al minimo le comunicazioni tra client e sistema. Un'ulteriore caratteristica di questo layer è l'essere *stateless*, cioè ciascuna chiamata a un metodo del Service layer non deve riferirsi a precedenti chiamate a esso. Ogni stato tra chiamate di metodi va tenuto all'interno del Domain Model layer. Mantenedo tali principi si ottiene un buon incapsulamento di tutti i casi d'uso del sistema. Diventa, quindi, anche più facile aggiungere meccanismi di comunicazione diversi a una singola interfaccia di sevizio. Si ottiene, così, una migliore adesione al principio DRY[2] e al conseguente beneficio in fatto di riutilizzo del codice.

### Domain Model layer

Nel Domain Model layer si colloca la *business logic* dell'applicazione. Questo layer contiene la logica degli stati e dei casi di uso. Esso raccoglie gli oggetti da manipolare con il loro stato ed il loro comportamento. Se esso non contenesse tali informazioni, si andrebbe incontro alla costruzione di un cosidetto *Anemic Domain Model*. Si farà abbondante uso di tutte le regole di OOP (Object Oriented Programming) come polimorfismo ed ereditarietà. Si sottolinea che se nel Service layer si è incoraggiati a fare uno spinto utilizzo di iterfacce, nel Domain Model layer tale utilizzo, sebbene sempre apprezzabile, deve essere realizzato secondo necessità, evitando, quindi, un ploriferare di interfacce non utili. Il significato del termine *business logic* è ampio: si tratta di ogni regola o specifica richiesta dal committente e spazia da un complesso sistema di controlli di stato a una semplice regola di validazione dei dati inseriti. Si noti che, in alcuni casi, tale logica può essere affidata all'esterno del layer, andando in deroga al principio di incapsulamento di tali servizi all'interno del Domain Model layer. Per fare un

[2]DRY (Don't Repeat Yourself), anche conosciuto come "Single Point of Truth", è un principio secondo il quale l'informazione non debba essere ripetuta e ridondante e non si debba esprimere lo stesso concetto più di una volta, specie se in forma diversa. Di particolare importanza, nell'informatica, diventa un Design pattern della programmazione secondo il quale bisogna evitare il più possibile la duplicazione del codice, poiché questa complica la mantenibilità e la leggibilità del codice stesso. DRY è uno dei concetti fondamentali espressi nel libro *The Pragmatic Programmer*. Un codice DRY riduce al minimo le informazioni ridondati e le duplicazioni, risulta molto più pulito, mantenibili e leggibile e ricorre, dove possibile, all'utilizzo di funzioni per accorpare in un unico punto funzionalità usate più volte.

esempio, un controllo di unicità di un campo all'interno del database conviene sia affidato al DBMS, pena un notevole calo delle perfomance. Si deve avere, però, cura nel gestire le eccezioni dovute a eventuali errori di accesso alla base di dati, non semplicemente mandando all'utente un messaggio di errore, ma specificandone il tipo, così da rendere più comprensibile l'utilizzo dell'applicazione. Infine, il Domain Model layer non deve avere alcuna dipendenza dai framework e dai container, così che possa essere testato al di fuori di essi.

#### Persistence layer

Il Persistence layer è responsabile dell'interfacciamento con il meccanismo preposto alla persistenza dei dati al fine di ritrovarli o salvarli. Tipicamente questo layer contiene i metodi CRUD (Create Read Update Delete). Il beneficio di isolare questo layer è quello di proteggere il sistema dai cambiamenti. Supponiamo, ad esempio, di voler passare dall'utilizzo di un DBMS ad un altro. Se non si fosse isolata in tale layer la logica di persistenza dei dati, si dovrebbe spendere parecchio tempo nel cercare e modificare le classi del sistema che hanno accesso al database; inoltre andrebbero ripensate le classi di test. Tale isolamento consente, inoltre, di testare l'applicazione independentemente dal database, accelerando considerevolmente i tempi di sviluppo. Certamente non si può fare a meno di verificare che il sistema funzioni con il database di utilizzo ma, come si vedrà in seguito, si tratta di integration testing.

### 9.1.2 Breve storia delle Web Application

Inizialmente i siti Web erano del tutto statici, in quanto presentavano delle semplici pagine `HTML`. Inoltre, il protocollo servente, ovvero l'`HTTP` (Hypertext Transfer Protocol), era un semplice protocollo *stateless*. Questa situazione non durò a lungo. Ben presto, infatti, nacque l'esigenza di mostrare nei siti delle informazioni che potessero cambiare nel tempo. Le persone, inoltre, desideravano compiere operazioni sempre più complesse con l'utilizzo dei siti Web, per esempio tenere traccia di ciò che un utente aveva fatto all'ultimo accesso al sito, in modo da consentire relazioni commerciali, come aggiungere oggetti ad un carrello. Si necessitava, quindi, di un meccanismo che fornisse contenuti dinamici e che desse la possibilità di memorizzare uno stato con un protocollo *stateless* come `HTTP`.

#### Script CGI: il primo meccanismo per i contenuti dinamici

Il primo meccanismo che mise a disposizione degli utenti contenuti dinamici è stata la Common Gateway Interface (`CGI`). Applicazioni eseguibili (di solito, ma non necessariamente, scritte in `Perl` o `C`) venivano fornite con un'interfaccia che abilitava i client ad accedervi attraverso il protocollo `HTTP`. Nonostante tutto, `CGI` soffriva di numerosi svantaggi:

- Ogni richiesta `CGI` necessitava dell'avvio di un processo del sistema operativo.
- Ognuno dei processi creati, quindi, caricava ed eseguiva un programma `CGI`.
- Una codifica noiosa e ripetitiva era necessaria per gestire il protocollo di rete e per interpretare le richieste.

Le prime due operazioni dell'elenco precedente potevano utilizzare un gran numero di cicli della CPU ed una grande mole di memoria. Poichè entrambe le operazioni dovevano essere eseguite per ogni richiesta, vi era la possibilità che un server si sovraccaricasse nel caso in cui esso ricevesse un gran numero di richieste in un breve lasso di tempo. Inoltre, poichè i programmi `CGI` erano mutuamente indipendenti, e spesso scritti in linguaggi tra loro incompatibili, non era possibile riutilizzare il codice per la gestione della rete e per l'interpretazione delle richieste. Si noti che `CGI` descriveva solo il contratto tra il server web e il programma. Non erano, infatti, servizi per contribuire a implementare un servizio *user-centric* (centralizzato sull'utente). Era, quindi, difficile memorizzare l'identità dell'utente attraverso le varie richieste, limitare l'accesso all'applicazione ai soli utenti autorizzati o memorizzare informazioni *runtime* nell'applicazione. Per questo motivo, gli script `CGI` non vengono utilizzati nei moderni siti Web.
Si è reso, quindi, necessario poter lavorare con un *framework* per la creazione di applicazioni Web in grado di fornire le funzionalità appena citate e di risolvere il problema delle scarse prestazioni della scalabilità.

### Java lato server: Servlet

Le `Servlet` sono porzioni di codice scritte in `Java` che hanno una forma definita e vengono richiamate per generare dinamicamente contenuti o eseguire alcune azioni. Le `Servlet` superano i problemi di `CGI` precedentemente menzionati in quanto:

- L'`overhead` dovuto all'avvio di un processo del sistema operativo per ogni richiesta è completamente azzerato. Una Java Virtual Machine (`JVM`), infatti, è attiva in background e tutte le richieste sono da essa gestite.
- Le classi `Java` vengono caricate dalla `JVM` per elaborare le richieste in entrata; se più di una richiesta richiede il medesimo trattamento, la classe già caricata può essere utilizzata per gestirla. Questo elimina l'overhead del caricamento delle classi per tuttie le richieste tranne la prima.
- Il problema della gestione dello stato al di sopra di un protocollo *stateless* come `HTTP` è risolto, come verrà successivamente spiegato.
- Il codice che gestisce il protocollo di rete e che decodifica le richieste in arrivo può essere condiviso da tutte le classi Java che elaborano le richieste.

Come detto, le *Servlet* hanno una forma ben definita: questo significa che tutte le *Servlet* implementano un'interfaccia chiamata `Servlet`, che definisce il loro ciclo di vita standard, ovvero una lista di metodi che verranno chiamati in modo prevedibile. L'inizializzazione è agevolata dall'uso del metodo `init()`. Tutte le risorse necessarie per la *Servlet*, insieme alle eventuali inizializzazioni che la servlet deve fare prima di poter servire le richieste dell'utente, sono fornite da questo metodo, che viene chiamato una sola volta per ogni istanza della *Servlet*. Ogni *Servlet* può gestire più richieste da più utenti. La prima volta che si effettua la richiesta del file, quest'ultimo viene compilato, creando una *Servlet*, che sarà archiviata in memoria (per servire le richieste successive); solo dopo questi passaggi viene elaborata la pagina `HTML` che viene mandata al browser. Ad ogni richiesta successiva alla prima, il server controlla se sulla pagina richiesta è stata effettuata qualche modifica, in caso negativo richiama la *Servlet* già compilata, altrimenti si occupa di eseguire nuovamente la compilazione e memorizzare la nuova. L'interfaccia `Servlet` definisce, inoltre, un metodo `service()` che viene chiamato ad ogni richiesta utente. Tale metodo controlla l'elaborazione e la generazione della risposta da mandare al *client*. Quando una richiesta viene servita, la *Servlet* resta in attesa della successiva richiesta. Il metodo `service()` si occupa, inoltre, di verificare che tipo di richiesta `HTTP` è stata fatta (per esempio `GET` o `POST`), ed inoltra la richiesta al metodo opportuno. Infine, un metodo chiamato `destroy()` viene invocato per "smaltire" la classe *Servlet* (vedi figura 9.3). Questo metodo si occupa, inoltre, di liberare le risorse acquisite dal metodo `init()`.

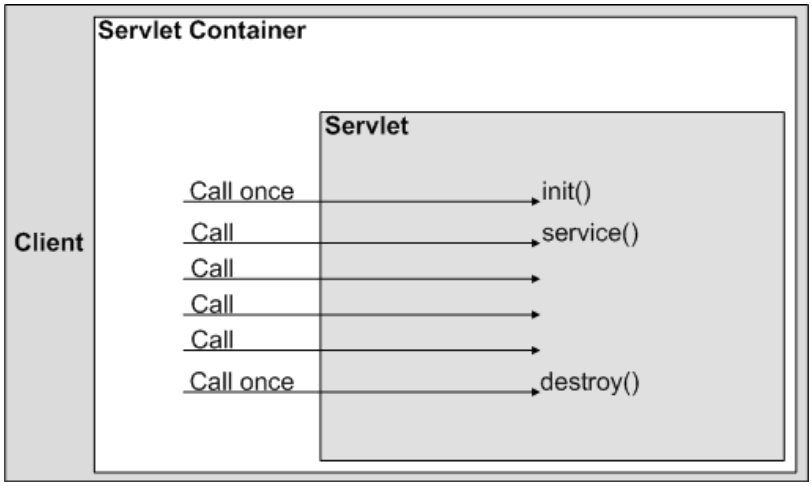


**Figura 9.3.** Metodi delle Servlet.

Molti fornitori sviluppano ambienti di esecuzione per le *Servlet* noti come *Servlet Container*. Ciascuno di essi deve garantire che vengano seguite le specifiche delle *Servlet*, in modo che una *Servlet* scritta secondo tali specifiche possa

essere eseguita in diversi ambienti senza dover apportare delle modifiche. I *containers* forniscono, inoltre, dei servizi in aggiunta a quelli per la gestione del ciclo di vita di una *Servlet*. Tra di essi si possono menzionare quelli che mettono a disposizione i parametri di inizializzazione, le connessioni con le basi di dati e quelli che permettono la gestione delle sessioni. Per risolvere il problema del mantenimento della sessione dell'utente, il *container* mantiene l'identità del client tramite *cookies* temporanei che memorizzano un *token* che fa riferimento all'utente. in questo modo il *container* è in grado di identificare un client attraverso più richieste.

Sebbene le *Servlet* offrano un notevole miglioramento rispetto a *CGI*, in particolare per quanto riguarda le prestazioni ed il carico del server, anch'esse presentano alcuni inconvenienti. Esse sono state ideate per l'elaborazione della logica, ma risultano meno utilizzabili per la presentazione dei contenuti (ad esempio per l'`HTML`). La codifica dell'output testuale nel codice, compresi i tag `HTML`, rende l'applicazione meno gestibile perchè, quando il testo in `HTML` deve essere cambiato, le *Servlet* devo essere ricompilate. In secondo luogo, è necessario che il designer `HTML` abbia delle buone conoscenze di `Java` per evitare di generare bachi nella *Servlet*. Spesso capita che il programmatore dell'applicazione riceva del codice `HTML` da un altro designer e che lo incorpori nel codice della *Servlet* in corso d'opera. Questo, però, è una procedura incline agli errori.

Per risolvere questo problema Sun Microsystems ha introdotto le *JavaServer Pages* note anche con l'acronimo *JSP*.

### JavaServer Pages

Le specifiche della prima edizione delle *JavaServer Pages* (*JSP*) presentavano molte somiglianze con *Active Server Pages* (*ASP*), una tecnologia Microsoft. Entrambe si sono, però, molto evolute da allora, tanto che la loro somiglianza è, oggi, molto ridotta. La tecnologia `JSP` è stata migliorata con l'introduzione delle *Tag Libraries*. Queste librerie di tag sono raccolte di tag personalizzati; ogni tag corrisponde ad un modulo `Java` altamente riutilizzabile.

Dietro le quinte, la `JSP` viene compilata in una classe *Servlet* la prima volta che essa viene invocata. Questa *Servlet* viene, quindi, richiamata ad ogni successiva richiesta, evitando, perciò, l'analisi e la compilazione della `JSP` ogniqualvolta l'utente accede al sito. Le `JSP` si diffusero notevolmente grazie alla loro grande idoneità alla creazione di contenuti visuali dinamici, in un momento in cui Internet moltiplicava la sua popolarità. Come le *Servlet*, anche le `JSP` operano all'interno di un *container*. Il *JSP container* offre gli stessi servizi di un *Servlet Container*, ma richiede, in aggiunta, il passo supplementare per la conversione in *Servlet* della `JSP` e per la compilazione del codice prima della sua esecuzione. Come si vedrà nelle sezioni successive, Apache Tomcat contiene sia il *Servlet container* (chiamato Catalina), che esegue le *Servlet* e le `JSP` compilate, sia il compilatore per le `JSP` (chiamato Jasper). La combinazione di un compilatore per `JSP` ed un *Servlet container* prende il nome di *Web Container* (un conteni-

tore in grado di ospitare applicazioni Web `Java`). Una differenza pratica tra le *Servlet* e le `JSP` sta nel fatto che le prime sono fornite già compilate, cosa che non avviene necessariamente per le seconde. Per un amministrazione di sistema questo significa che i file delle *Servlet* devono essere memorizzati in un'area riservata del server, mentre le pagine `JSP` possono tranquillamente essere mescolate alle pagine `HTML` statiche, alle immagini e alle altre risorse della sezione pubblica. In ogni caso, se non viene seguita una buona politica di sviluppo, questo può condizionare la mantenibilità di un sito.

Con l'avvento delle *Servlet* e delle `JSP`, sono state sviluppate moltissime applicazioni Web risultate, poi, poco mantenibili. Le cause di tutto ciò erano le seguenti:

- il flusso di controllo dell'applicazione Web, ovvero quali contenuti devono essere visualizzati e in quale ordine, era spesso codificato all'interno delle pagine Web stesse;
- la *business logic* del sito Web era strettamente associata alla presentazione dell'interfaccia utente.

Questa tipologia di architetture è, oggi, nota come *Model 1 architecture*. Essa si adatta solamente a siti di piccole dimensioni e con funzionalità limitate, o con pagine Web con limitati requisiti di espansione. Realizzare siti Web in questo modo è molto semplice e, quindi, la produttività aumenta quando la complessità è bassa. Questo modello non è, però, raccomandato per siti di maggiori dimensioni, in quanto il tempo risparmiato in fase di sviluppo verrebbe poi perso in fase di debug. La struttura di un'applicazione Web con architettura Model 1 è rappresentata in figura 9.4. Osservando la figura, è semplice capire perchè tale approccio può presto diventare ingestibile con l'aumento della complessità del sito Web e rende difficili anche i cambimenti nel controllo del flusso. Come detto in precedenza, uno dei problemi più frequenti consisteva nel mischiare logica dell'applicazione con presentazione dei dati. L'alternativa a tutto ciò consiste nel mantenere le pagine esenti da codice `Java` (ove possibile), e localizzare la logica in classi `Java`. Lo sviluppo delle moderne applicazioni Web include alcune regole di base che rendono i siti e le applicazioni più semplici da gestire ed, eventualmente, da estendere:

- Non incorporare la logica della gestione delle richieste degli utenti ed il flusso di controllo all'interno delle stesse pagine `JSP`. Questo rende difficile la manutenzione del codice.
- Non mescolare la logica applicativa con la logica dell'interfaccia utente (meglio nota come logica di presentazione). L'architettura MVC, che verrà illustrata più avanti, spiega come questo è fatto. Questa architettura è, talvolta, chiamata *Model 2 architecture* in contrasto con quella *Model 1* menzionata in precedenza.

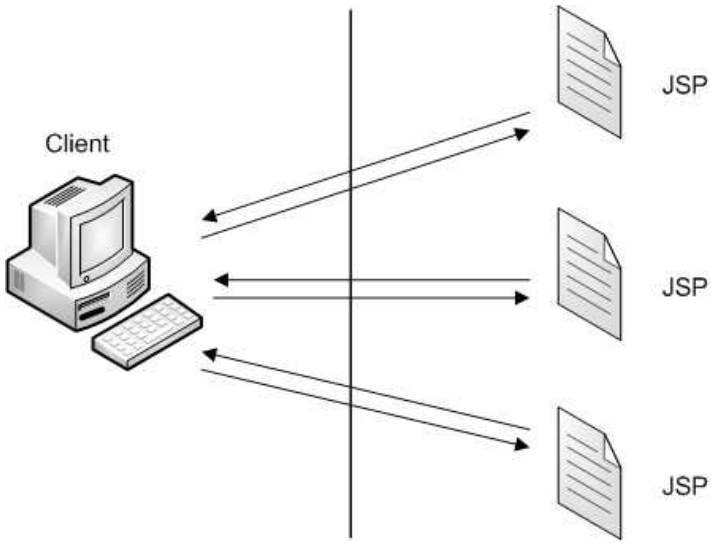


**Figura 9.4.** Struttura di un'applicazione Web con architettura Model 1.

- Tenere il codice `Java` fuori dalle pagine `JSP`. Le *Tag libraries* aiutano gli sviluppatori a fare tutto ciò.

- Un'altra buona pratica è quella di usare le pagine `JSP` come dei template. Per esempio l'intestazione della pagina che include, per ipotesi, il logo dell'azienda può trovarsi in una pagina `JSP`, il menu principale del sito in un seconda e l'informazione da visualizzare in una terza. Quando l'utente effettua la richiesta, questi elementi separati vengono assemblati e visualizzati dall'utente come un'unica pagina.

### 9.1.3 Architettura MVC

Nella sezione precedente è stato mostrato un esempio di applicazione con architettura *Model 1*, in cui le diverse pagine `JSP` devono sapere a quale pagine indirizzare l'utente e da quale esso proveniva. Tutto ciò diventa presto molto complicato e difficile da gestire per applicazioni sufficientemente complesse. L'architettura *Model 2* o *MVC* (*Model View Controller*) aiuta a risolvere il problema permettendo la separazione della logica applicativa dalla presentazione del contenuto `HTML`. Il *Model* corrisponde alla logica dell'applicazione, ovvero alle regole che determinano ciò che viene mostrato e a chi. Il componente *View* è dato, invece, dall'insieme delle pagine (nel nostro caso `JSP`) che visualizzano il contenuto creato. Il *Controller*, infine, determina quale parte del Model viene invocato e quale pagina `JSP` viene caricata per effettuare il rendering dei dati. In altre parole, il Controller definisce la struttura dell'applicazione e la logica del flusso delle pagine. Queste informazioni vengono lette da dei file di configu-

razione e non sono, quindi, "sepolte" nel codice delle pagine Web, a differenza di quanto avveniva con le applicazioni *Model 1*. La figura 9.5 mostra lo schema dell'architettura MVC.

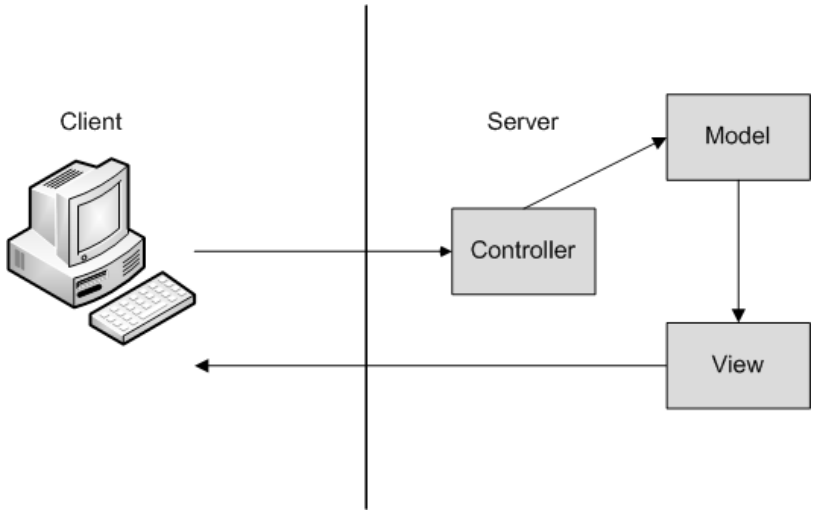


**Figura 9.5.** Struttura di un'applicazione Web con architettura MVC.

### 9.1.4 La struttura dell'applicazione WQF

Quanto emerso dall'analisi funzionale e dai colloqui con l'azienda, ha portato ad individuare nell'MVC un *pattern di progettazione* (*design pattern*[3]) quale punto di partenza nello sviluppo di un software in grado di soddisfare i requisiti espressi. Prima che il pattern MVC venisse definito, le interfacce grafiche tendevano a "mischiare" questi tre oggetti.

L'utilizzo di questo pattern permette, quindi, di disaccoppiare memorizzazione, manipolazione e presentazione dei dati. L'effetto finale è quello di permettere uno sviluppo e una gestione separata dei vari blocchi che compongono un software.
L'applicazione di questo design pattern nell'ambito dello sviluppo di un software per il sistema di gestione della qualità risulta molto interessante. È importante sottolineare la necessità di utilizzare un supporto informativo che sia completamente indipendente dalle viste presentate ai vari attori. Tutto ciò permette la massima integrazione con strumenti software già in uso, senza la necessità di dover cambiare il metodo con cui questi vengono gestiti ed utilizzati. È, infatti, esclusivo compito della porzione Control del software effettuare la traduzione tra dati grezzi e viste (o viceversa). Tra l'altro, Control non è necessariamente vincolato ad un unico modello di presentazione dei dati, ma può essere facilmente esteso per supportare diverse viste.

La figura 9.6 mostra lo schema completo dell'architettura MVC per l'applicazione WQF. Nel disegno sono, inoltre, indicate le tecnologie utilizzate, le quali verranno approfondite nel prossimo capitolo.

## 9.2 Tecnologie e software utilizzati

### 9.2.1 Il linguaggio di programmazione: `Java`™

`Java`[4] fu concepito da James Gosling, Patrick Naughton, ChrisWarth, Ed Frank e Mike Sheridan in Sun Microsystems Inc. nel 1991 e richiese 18 mesi per sviluppare la prima versione funzionante. Inizialmente il linguaggio fu battezzato "OAK" ma venne rinominato "Java" nel 1995. Tra la versione iniziale di Oak nell'autunno 1992 e l'annuncio al pubblico di `Java` nella primavera del 1995, molte altre persone avevano contribuito al progetto e all'evoluzione del linguaggio. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin e Tim Lindholm diedero un contributo importantissimo per la maturazione del prototipo originale. La spinta iniziale per `Java` non venne da Internet, bensì dal bisogno di un linguaggio

[3] Nell'ingegneria del software, un design pattern (struttura di progettazione) può essere definito "una soluzione progettuale" generale a un problema ricorrente. Esso non è una libreria o un componente di software riusabile, quanto una descrizione o un modello da applicare per risolvere un problema che può presentarsi in diverse situazioni durante la progettazione e lo sviluppo del software. La differenza tra un algoritmo e un design pattern è che il primo risolve problemi computazionali, mentre il secondo è legato agli aspetti progettuali del software.

[4] `http://java.sun.com`

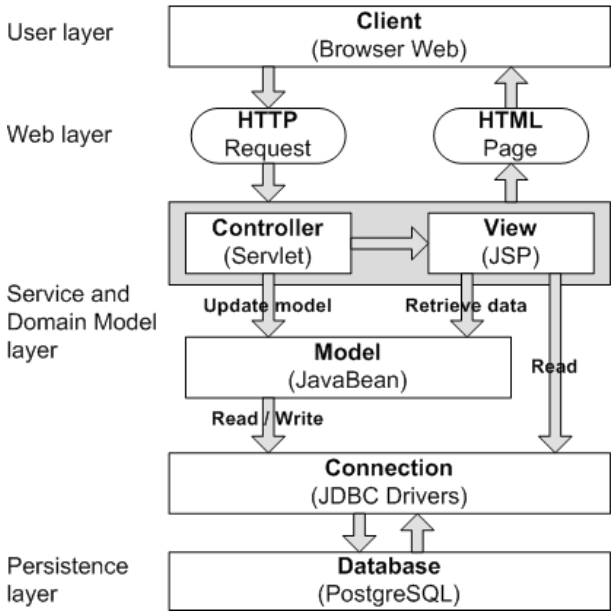


**Figura 9.6.** Schema completo dell'architettura MVC per l'applicazione WQF.

indipendente dalla piattaforma, cioè neutro rispetto all'architettura, che avrebbe potuto essere impiegato per creare software da incorporare in diversi dispositivi elettronici commerciali come forni a microonde e telecomandi.

`Java` è un linguaggio di programmazione *general-purpose*, concorrente, basato su classi e orientato agli oggetti; deriva dal `C` e dal `C++` ma è organizzato in modo differente, con molti aspetti omessi e nuove idee prese da altri linguaggi. Ecco i punti di forza di questo linguaggio:

- *semplice*: è stato concepito per essere semplice da apprendere, efficace da usare e rivolto ai programmatori professionisti; pur dovendo molto a `C++`, `Java` risulta essere molto più semplice e tollerante grazie anche alla mancanza di puntatori;

- *orientato agli oggetti*: implementa questo paradigma in modo esteso con l'eccezione, per ragioni di efficienza, dei tipi semplici di dati;

- *robusto*: è fortemente tipizzato, quindi controlla il codice già durante la compilazione; inoltre non c'è la necessità di dover liberare la memoria quando si è terminato di usare un oggetto perchè questo compito è demandato al *garbage collector*; anche la gestione delle eccezioni è orientata agli oggetti;

- *multiprocesso*: il sistema *run-time* di `Java` fornisce una soluzione elegante,

anche se sofisticata, per la sincronizzazione di multiprocessi che mette in grado di costruire sistemi interattivi funzionanti e affidabili;

- *indipendente dall'architettura*: l'obiettivo dei progettisti era "*scrivi una volta per tutte, esegui ovunque, in ogni momento, per sempre*" e per gran parte è stato raggiunto;

- *interpretato ad alte prestazioni*: `Java` consente la creazione di programmi multipiattaforma attraverso la compilazione in una forma intermedia chiamata bytecode; questo codice è stato pensato per ottenere alte prestazioni e può essere interpretato su ogni sistema che disponga di una Java Virtual Machine;

- *grande supporto*: `Java` dispone di varie API (*Application Program Interface*) per quasi ogni ambito, molto utili per lo sviluppo di applicazioni.

Lo scotto maggiore per questa serie di vantaggi lo si paga sul lato delle prestazioni visto che, a differenza dei linguaggi compilati, `Java` viene interpretato al momento dell'esecuzione.

### 9.2.2 Il Web Container: Apache Tomcat

Ai tempi della definizione delle specifiche `JSP 1.0` e `Servlet 2.0`, la Sun sviluppò il suo *Java Server Web Development Kit poi* rinominato *Java Servlet Development Kit* (*JSDK*), che utilizava un Web Server interamente realizzato in `Java`. Nel frattempo, un gruppo esterno di sviluppatori Open-Source, l'*Apache Java Group*, stava lavorando su un motore `JSP/Servlet` che fosse compatibile con le ultime API dei servlet e delle `JSP` ma che avrebbe dovuto integrarsi con i server Apache largamente diffusi; il motore prese il nome di *Apache JServ*. Realizzarono *JServ* in due parti: la prima era scritta in `C` e doveva fungere da modulo caricabile per il server Apache, l'altra era una implementazione in `Java`, di un *servlet container standalone*. Le due parti comunicavano per mezzo di un protocollo privato. Mentre il progetto *JSDK* della Sun era focalizzato sull'adesione alle specifiche, l'*Apache JServ* si concentrò sulle performance e sugli aspetti più pragmatici. Diventò presto evidente che i due progetti si sovrapponevano e che gli sforzi si sarebbero dovuti unire. Nel 1999 la Sun donò alla Apache Software Foundation il codice sorgente dell'implementazione di riferimento per le `JSP` e le `Servlet`; per integrare i due progetti sorse il gruppo di collaborazione *Jakarta* che finì, però, per includere tutti i progetti *open-source* dell'*Apache Java Group*. Tomcat[5] è uno di questi progetti: la serie 3.x discende direttamente dal progetto della Sun mentre la versione 4 impiega una nuova architettura ad alte prestazioni. L'attuale release di Tomcat è la 6.0 che implementa le specifiche per `Servlet 2.5` e `JSP 2.1`.

[5] `http://tomcat.apache.org`

Tomcat è un *Web Server* completamente scritto in `Java`. Più precisamente, si tratta di un *servlet container standalone* e rappresenta l'implementazione di riferimento per le tecnologie `JSP` e `Servlet`, con l'aggiunta di alcune caratteristiche che lo rendono più efficiente e ne fanno un'utile piattaforma per sviluppare e far girare applicazioni Web. Il servlet container gira su una Java Virtual Machine e può essere affiancato da un server Web in tre modi:

- *in-process*: il servlet container è legato al server Web da un *plug-in* che fa un po' da mediatore tra i due; *plug-in* e container si trovano nello stesso spazio di memoria del server così come la JVM che li esegue; questa configurazione garantisce le massime prestazioni dovute alla condivisione della memoria ma, d'altro canto, limita la scalabilità e l'affidabilità (il crash di un qualsiasi thread può portare al crash dell'intero server);
- *out-of-process*: a differenza del tipo precedente, qui si usano due spazi di memoria distinti: in uno gira il server con il *plug-in* `Java`, nell'altro si trova la JVM con il *servlet container*; solitamente plug-in e *container* comunicano usando il protocollo TCP-IP;
- *stand-alone*: in questo caso il *servlet container* funge da server Web vero e proprio e risponde direttamente alle richieste dei clients.

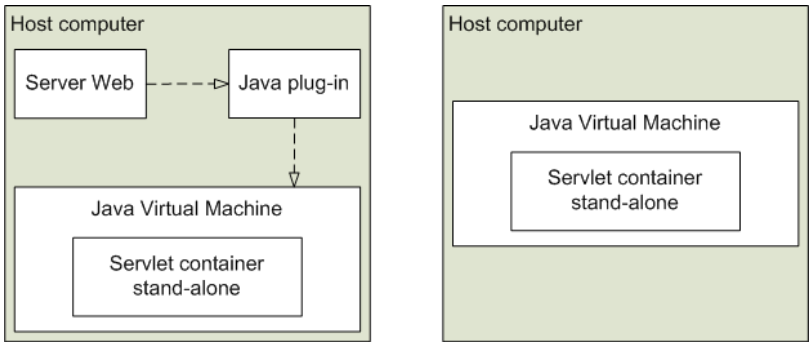


**Figura 9.7.** Un servlet container di tipo *out-of-process* (a sinistra) ed uno di tipo *stand-alone* (a destra).

La figura 9.7 mostra la struttura dei servlet container di tipo *out-of-process* e *stand-alone*.

Tomcat fornisce un servlet container di tipo *stand-alone*. Quest'ultimo, infatti, mette a disposizione i servizi di rete necessari a ricevere le richieste, decodificarle in base al protocollo `HTTP` e inviare le risposte. Inoltre ospita e gestisce le servlet durante tutto il loro ciclo di vita. Il flusso dei messaggi generato da una richiesta `HTTP` dell'utente (il client) si articola come segue:

- la richiesta viene ricevuta dal server Web e consegnata al *servlet container* che, come si è detto, può essere fisicamente separato;

- il *servlet container* determina, in base alla sua configurazione e ai parametri della richiesta, a quale servlet dovrà passare il controllo;
- il servlet usa l'oggetto che rappresenta la richiesta `HTTP` per individuare l'utente remoto e ricavarne il contenuto; esegue le operazioni per cui è stato programmato e prepara i dati da inviare al client;
- non appena il controllo ritorna al *servlet container*, questo si assicura che la risposta venga interamente inviata al client.

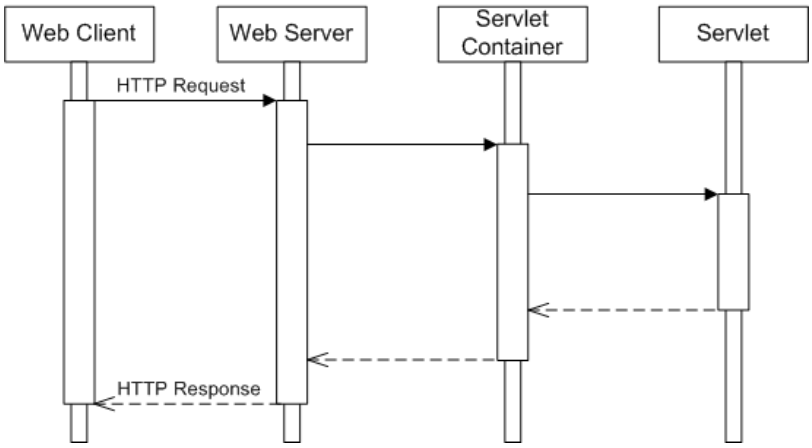


**Figura 9.8.** Flusso dei messaggi in un server Web dotato di un *servlet container* `Java`.

I parametri di configurazione e di inizializzazione di Tomcat, come quelli di ogni singola applicazione Web installata, vengono specificati da alcuni file in formato XML:

- `server.xml` viene letto ad ogni avvio del servlet container e contiene informazioni come il nome del server, la porta per la connessione `HTTP`, la directory di installazione delle applicazioni, i timeouts;
- `web.xml` può essere presente nella directory /WEB-INF/ di ciascuna applicazione e specifica:
  - quale meccanismo di autenticazione usare;
  - i parametri di inizializzazione dei servlet;
  - eventuali filtri che mappano le richieste `HTTP` e le inoltrano al servlet corretto;
  - eventuali pagine di errore;
  - la configurazione delle sessioni di lavoro;
  - l'uso di eventuali librerie di tag personalizzati.

### 9.2.3 I Java Beans

Come si è accennato non è molto comodo e conveniente inserire troppo codice `Java` (cioè *scriptlet*) all'interno delle pagine `JSP`. Può, invece, essere molto utile racchiuderlo in una classe apposita che viene chiamata al momento del bisogno; per questo esistono i `JavaBeans`[6] ovvero l'architettura a componenti di `Java`. I vantaggi offerti da questi componenti software nell'ambito delle applicazioni Web sono quelli di permettere la riusabilità del software (*Write Once Run Anywhere*, ovvero "scrivi una volta, esegui dappertutto"), la separazione tra contenuto e codice, l'implementazione di uno stato dell'applicazione (compensando la natura "senza stati", *stateless*, del protocollo `HTTP`). La specifica relativa ai `JavaBeans` è molto complessa, ma vale la pena di elencarne alcune caratteristiche:

- non devono avere alcuna proprietà pubblica;
- le proprietà che devono essere "esposte" devono avere metodi `set` e `get` secondo necessità;
- devono avere almeno un metodo costruttore senza argomenti per poter caricare il bean a piacimento.

Le `JSP` mettono a disposizione alcune direttive per accedere alle proprietà dei beans.

### 9.2.4 Spring Framework

Affrontare la progettazione di un'applicazione comporta la scelta quasi obbligata di un *framework*. Sebbene si possa pensare di poter realizzare da soli ogni aspetto del proprio lavoro, è saggio ricorrere a soluzioni già sperimentate e ampiamente testate offerte da numerosi progetti open source che forniscono gli strumenti di base per lo sviluppo veloce del proprio codice. La scelta è, fortunatamente, molto ricca. Probabilmente il *framework* di maggior successo è *Spring Framework*[7] e, questo, è stato scelto per lo sviluppo del progetto WQF. Il principio guida di tale *framework* è quello di essere leggero; tale leggerezza non si riferisce al numero di classi o alle dimensioni della distrubuzione, quanto, piuttosto, al suo minimo impatto. Ottenere i benefici del nucleo fondamentale di *Spring Framework* richiede, infatti, minimi cambiamenti nel codice della propria applicazione. Il nucleo principale appena citato di *Spring Framework* è basato sull'*Inversion of Control* altrimenti detto *Dependency Injection*; si tratta di una tecnica che estrae la creazione e la gestione delle dipendenze dei componenti. Si consideri, ad esempio, una classe `Alpha` che dipende da un'istanza della classe `Beta` per realizzare un determinato compito; tipicamente `Alpha` dovrebbe creare un'istanza

[6] `http://java.sun.com/javase/technologies/desktop/javabeans/`
[7] `http://www.springsource.org/`

di `Beta` utlizzando l'operatore `new` o una classe `factory`; l'approccio della *Dependency Injection* permette di fornire l'istanza necessaria a `Alpha` in *runtime*. I benefici di tale approccio si possono sintetizzare nei seguenti punti:

**Ridurre il codice di legame** : uno dei punti di maggior valore della *Dependency Injection* è la capacità di ridurre il codice da scrivere per legare i differenti componenti di un'applicazione. Spesso ciò può essere banale riducendosi alla creazione di una nuova istanza, ma ciò può risultare complesso nel caso di risorse remote.

**Estrarre le dipendenze** : ciò consente di passare dall'implementazione di una dipendenza a un'altra più facilmente.

**Gestire le dipendenze in un unico luogo** : un'applicazione di una certa complessità ha tipicamente le sue dipendenze diffuse su tutto il codice che la costituisce. Usando la *Dependency Injection* tutte le informazioni riguardanti le dipendenze sono contenute in un unico luogo rendendo la gestione di tali dipendenze meno problematica.

**Aumentare la testabilità** : programmando una classe in funzione della *Dependency Injection* implica la possibilità di sostituire le dipendenze facilmente, aspetto particolarmente utile in fase di test. Se un'applicazione necessita dell'accesso a un database, ma si vuole fare un test indipendentemente da esso, sarà sufficiente creare un'implementazione fittizia della dipendenza dal database che ritorni una serie di dati da passare all'oggetto sotto test; tale meccanismo favorisce anche il test di applicazioni web, basterà usare dei *mock object*[8] di `HttpServletRequest` e `HttpServletResponse`.

**Promuovere una buona progettazione** : progettare per la *Dependency Injection* significa, essenzialmente, programmare per interfacce: tipicamente i maggiori componenti sono definiti da interfacce le cui concrete implementazioni sono create e collegate dalla *Dependency Injection*.

Numerose sono le possibilità offerte da *Spring Framework* come si può vedere dall figura 9.9. Innanzitutto, il supporto dell'AOP (*Aspect Oriented Programming*) fornisce la possibilità di realizzare una logica di controllo trasversale, cioè applicabile a più parti di un'applicazione automaticamente. Vi sono due principali tipi di AOP:

- statico: fornisce la possibilità, durante la compilazione, di costruire la logica AOP con cui arricchire l'applicazione;

[8] Nella programmazione orientata agli oggetti, i *mock objects* sono oggetti simulati che imitano il comportamento degli oggetti reali in modo controllato. Un programmatore di computer crea, in genere, un oggetto fittizio per testare il comportamento di qualche altro oggetto, più o meno allo stesso modo in cui il progettista di una vettura utilizza un manichino da crash test per simulare il comportamento dinamico di un umano negli impatti del veicolo.

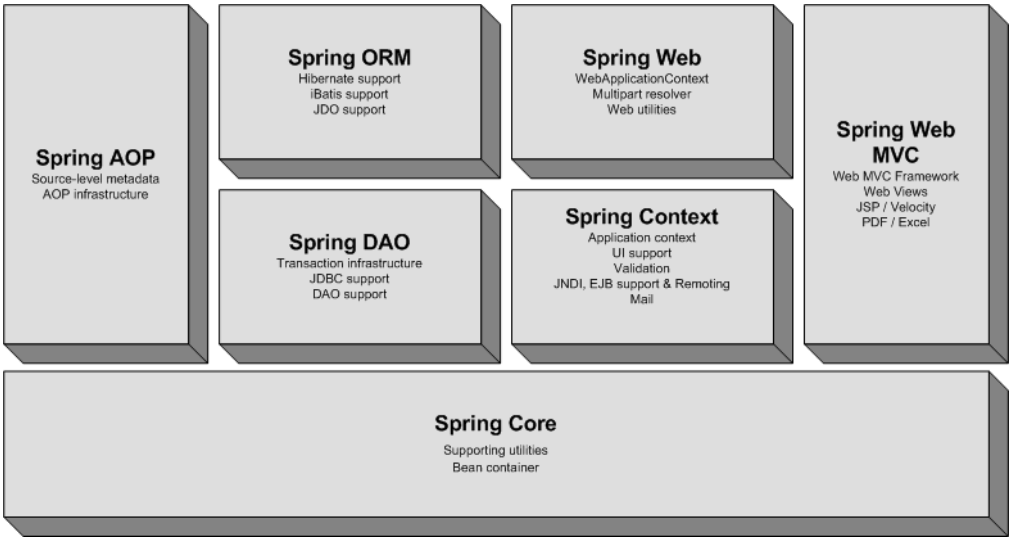


**Figura 9.9.** Struttura di *Spring Framework*.

- dinamico: è il caso di *Spring Framework* in cui la logica trasversale è applicata in *runtime*.

*Spring Framework* supporta anche numerose tecnologie DAO (*Data Access Object*) che possono essere integrate simultaneamente. Oltre alla JDBC (*Java DataBase Connectivity*), *Spring Framework* consente l'utilizzo di diverse tecnologie di ORM (*Object Relational Mapping*) tra cui Hibernate, JDO (*Java Data Objects*), Oracle TopLink, OJB (*Apache Jakarta's ObJect Relational Bridge*), iBATIS SQL e JPA (*Java Persistence API*). *Spring Framework* consente, inoltre, di relizzare transazioni sia dichiarative che programmatiche. Sebbene questo framework consenta lo sviluppo di applicazioni desktop, esso ha un supporto particolarmente ricco per le web application di cui si tratteranno vari aspetti nel corso della presente tesi. Un'altro ricco settore di sviluppo con *Spring Framework* è quello del RMI (*Remote Method Invocation*) di cui supporta `Java` RMI, Caucho Hessian, Caucho Burlap e, oltre a tali protocolli, anche un protocollo basato su `HTTP` che sfrutta la serializzazione di Java™. Un altro campo in cui si distingue questo framework è la gestione dell'invio di email, esso fornisce ricche API e la possibilità di configurare due implementazioni: `JavaMail` e `MailMessage`. Molte applicazioni spesso necessitano di operazioni di avvicendamento come l'invio di informazioni a clienti o l'avvio, in dati momenti, di funzionalità dell'applicazione; anche in questo campo *Spring Framework* offre diverse possibilità: una utilizza la classe `Timer` di `Java`™, l'altra `Quartz Scheduler`. Sebbene vi siano ancora diverse funzionalità supportate da *Spring Framework* si conclude questa breve presentazione citando l'opportunità, offerta dallo stesso, di evitare la scrittura di molto codice per gestire le eccezioni fornendo, allo stesso tempo, una gerarchia

di eccezioni veramente granulare.

### 9.2.5 L'ambiente di sviluppo: Eclipse

*Eclipse*[9] è un progetto open source legato alla creazione e allo sviluppo di una piattaforma di sviluppo ideata da un consorzio di grandi società quali *Ericsson*, *HP*, *IBM*, *Intel*, *MontaVista Software*, *QNX*, *SAP* e *Serena Software*, chiamato *Eclipse Foundation*, e creata da una comunità strutturata sullo stile dell'open source. Pur essendo orientato allo sviluppo del progetto stesso, questo IDE (*Integrated Development Environment*, in italiano "ambiente di sviluppo integrato") è utilizzato anche per la produzione di software di vario genere. Si passa infatti da un completo IDE per il linguaggio `Java` (JDT, *Java Development Tools*) ad un ambiente di sviluppo per il linguaggio `C++` (CDT, *C/C++ Development Tools*) e a plug-in che permettono di gestire `XML`, `PHP` e persino di progettare graficamente una GUI per un'applicazione `Java` (Eclipse VE, *Visual Editor*), rendendo di fatto *Eclipse* un ambiente RAD. Il programma è scritto in linguaggio `Java`, ma anziché basare la sua GUI su Swing, il toolkit grafico di Sun Microsystems, si appoggia a `SWT`, librerie di nuova concezione che conferiscono ad Eclipse una straordinaria reattività. La piattaforma di sviluppo è incentrata sull'uso di plug-in, delle componenti software ideate per uno specifico scopo, per esempio la generazione di diagrammi `UML`, ed in effetti tutta la piattaforma è un insieme di plug-in, versione base compresa, e chiunque può sviluppare e modificare i vari plug-in. Nella versione base è possibile programmare in `Java`, usufruendo di comode funzioni di aiuto quali: completamento automatico (*Code completion*), suggerimento dei tipi di parametri dei metodi, possibilià di accesso diretto a CVS e riscrittura automatica del codice (funzionalità questa detta di *Refactoring*) in caso di cambiamenti nelle classi. Essendo scritto in `Java`, *Eclipse* è disponibile per le piattaforme Linux, HP-UX, AIX, Mac OS X e Windows.

### 9.2.6 Il DBMS: PostgreSQL

*PostgreSQL*[10] ebbe inizio come *Ingres*, sviluppato all'università di Berkeley in California tra il 1977 e il 1985. Il sorgente di *Ingres* fu preso e migliorato dalla *Relational Technologies/Ingres Corporation* che produsse i primi server database che ottennero successo commercialmente (*Ingres Corp.* fu, in seguito, acquistata dalla *Computer Associates*). Nel 1986, a Berkeley, Michael Stonebraker capeggiò un gruppo per sviluppare un database server chiamato *Postgres* (1986-1994) sponsorizzato dalla *Defense Advanced Research Projects Agency* (*DARPA*), dall'*Army Research Office* (*ARO*), dalla *National Science Foundation* (*NSF*), e da *ESL Inc.*. Il sorgente di Postgres fu preso dalla *Illustra* e sviluppato come prodotto commerciale (l'*Illustra* fu più tardi comprata da *Informix* e integrata nel *Informix's Universal Server*). Due studenti laureati a Berkeley, Jolly Chen e

[9] `http://www.eclipse.org/`
[10] `http://www.postgresql.org/`

Andrew Yu, aggiunsero l'interfaccia SQL a *Postgres*, e lo chiamarono *Postgres95* (1994-1995). Essi lasciarono Berkeley, ma Jolly continuò ad aggiornare e supportare *Postgres95*, che ha ancora una mailing list attiva. Nell'estate 1996, divenne chiaro che la domanda di un database server SQL con tecnologia Open Source era forte e che, quindi, si doveva formare un team per continuare lo sviluppo. Marc G. Fournier a Toronto (Canada) si offrì di ospitare la mailing list e approntò un server per ospitarne i sorgenti. Il migliaio di iscritti alla mailing list furono spostati sul quella nuova ed il server fu configurato, fornendo alle persone gli account per apportare le modifiche ai files sorgenti usando *CVS*. Le persone maggiormente impegnate furono: Marc G. Fournier, Thomas Lockhart da Pasadena (California), Vadim Mikheev da Krasnoyarsk (Russia), e Bruce Momjian. Il loro primo obiettivo fu esaminare la vecchia mailing list, valutare le patch che erano state pubblicate per fisssare i vari problemi. Allora il sistema era molto fragile e non facilmente comprensibile. Pur parlando, già a quei tempi, di aggiungere delle caratteristiche, l'instabilità del sistema costrinse gli sviluppatori a focalizzarsi sulla soluzione dei problemi esistenti. Nel 1996 il team decise di cambiare il nome da *Postgres95* a *PostgreSQL* che evidenziava le doti SQL e rilasciarono le release ogni 3-5 mesi. L'ORDBMS *PostgreSQL* viene, oggi, distribuito sotto licenza BSD.

*PostgreSQL* è uno dei più avanzati database *Open Source* e la ricchezza delle sue funzionalità può essere paragonata a database commerciali quali *DB2* e *Oracle*. Di seguito si riporta una breve lista delle caratteristiche principali:

- **DBMS relazionale e a oggetti**: si tratta infatti di un database che oltre ad essere relazionale è anche a oggetti, cioè estende alla base di dati il paradigma di programmazione a oggetti;
- **elevata estendibilità**: l'utente può usare operatori, funzioni, metodi di accesso e tipi di dati definiti dall'utente stesso;
- **supporto completo di SQL**: *PostgreSQL* supporta le specifiche fondamentali di SQL99 e funzionalità avanzate di SQL92;
- **integrità referenziale**: *PostgreSQL* implementa l'integrità referenziale per garantire la validità dei dati nel database;
- **API flessibili**: tra queste interfacce sono comprese quelle per Object Pascal, Python, Perl, PHP, ODBC, JDBC, Ruby, TCL, C/C++ e Pike;
- **linguaggi procedurali**: supporta linguaggi procedurali interni come quello nativo PL/pgSQL e altri quali Perl, Python e Tcl/Tk;
- **Multi Version Concurrency Control (MVCC)**: tecnologia usata per evitare i lock non necessari. Normalmente l'utente in fase di lettura è bloccato da chi sta aggiornando i record. In *PostgreSQL* chi legge non viene mai bloccato da chi scrive perchè il database tiene traccia di tutte

le transazioni effettuate dagli utenti ed è in grado di gestire i record senza far attendere la loro disponibilità;

- **client/server**: *PostgreSQL* riserva un processo per ogni utente che cerchi di connettersi;

- **Write Ahead Logging (WAL)**: tecnologia che incrementa l'affidabilità registrando le modifiche prima che siano scritte nel database; ciò garantisce che, nel caso si verifichi un crash, esista un salvataggio delle transazioni con cui effettuare il rispristino.

### 9.2.7 Il sistema di versioning: Subversion

Qualsiasi progetto di dimensioni sufficientemente grandi raggiunge, alla fine, un punto in cui la comunicazione tra i vari partecipanti diventa un vero e proprio collo di bottiglia. Assicurarsi che ognuno abbia l'ultima versione di un documento importante, evitare la duplicazione degli sforzi e limitare la possibilità che qualcuno sovrascriva il lavoro cominciato da un altro possono sembrare semplici operazioni ma, man mano che la dimensione di un progetto cresce e, con essa, anche il numero di persone coinvolte, quello che una volta sembrava semplice diventa sempre più complesso e, di conseguenza, richiede più tempo e fatica. Nel mondo dello sviluppo software, uno dei modi per gestire queste complessità è l'utilizzo di sistemi di controllo di versione (in inglese version control systems). Un software di questo tipo costituisce un modo per gestire la complessità di lavorare con un team di persone su un progetto specifico. Esso consente, infatti, di memorizzare i risultati parziali del progetto (nel settore dello sviluppo software essi consistono nel codice sorgente e nella documentazione per il programma in corso di elaborazione) in una locazione centrale (per esempio un server aziendale) in modo che i vari membri del team abbiano sempre l'accesso alle ultime versioni dei file. Viene, inoltre, automatizzato il processo di aggiornamento della propria copia locale del progetto, includendo le ultime modifiche dei colleghi e, al contrario, vengono facilmente ridistribuite ad essi le nostre aggiunte. Infine, ma probabilmente cosa più importante, un software di versioning fornisce i dati storici sull'andamento del progetto, ovvero il monitoraggio di ogni variazione che è stata fatta nel processo dal momento della sua creazione. A prima vista questo può non sembrare così importante ma spesso, alla fine del ciclo di vita di un progetto, si desidera tornare indietro per vedere come e perché il progetto si è evoluto nella sua forma attuale, per cercare di individuare l'origine di un bug, per recuperare una qualche parte di lavoro che è stato inavvertitamente rimosso nella versione attuale, o, magari, per estrarre una particolare modifica che potrebbe essere applicata ad un'altra versione del progetto. In generale, quando si utilizza un sistema di controllo di versione, il lavoro si svolge nel seguente modo: per prima cosa si crea uno spazio per il progetto nel repository centrale, nel quale viene caricata la versione iniziale. Quindi, ogni componente del team di sviluppo

lavora sulla propria copia locale e, a mano a mano che vengono apportate delle modifiche, ricarica i file nel repository. Il repository tiene traccia della storia del progetto, memorizzando ogni modifica che è stata apportata.

Sistemi diversi utilizzano tecniche differenti al fine di garantire che le modifiche effettuate da un utente non entrino in conflitto con il lavoro degli altri utenti, non ci si deve, quindi, preoccupare che due persone modifichino la stessa parte di un progetto o sovrascrivano, accidentalmente, l'una il lavoro dell'altra. Poichè, come anticipato pocanzi, la cronologia delle modifiche viene mantenuta, è, di solito, abbastanza semplice tornare indietro nel tempo per cercare di capire perchè un dato cambiamento è stato fatto, il che risulta molto utile nel caso si abbiano problemi di debug.

Ciascuna di queste funzionalità e molto utile, ma è la loro combinazione a risultare davvero preziosa. Per questo motivo, una volta utilizzato un sistema di controllo della versione come strumento di lavoro, risulta complicato tornare a lavorare senza.

*Subversion* (noto anche come *SVN*, dal nome del suo client a riga di comando) è un software di versioning open source completamente gratuito. Esso può operare attraverso le reti, il che gli consente di essere utilizzato da computer di persone diverse. Alcuni software di versioning sono, allo stesso tempo, anche sistemi di software configuration management (SCM). Questi sistemi sono studiati espressamente per gestire gli alberi del codice sorgente, e hanno molte caratteristiche che sono specifiche per lo sviluppo software. Subversion, tuttavia, non è uno di questi sistemi. Si tratta, infatti, di un sistema generale che può essere utilizzato per gestire qualsiasi raccolta di file.

### TortoiseSVN

In tutti i sistemi operativi più comuni, esiste la possibilità di utilizzare dei programmi provvisti di interfaccia grafica atti a semplificare il più possibile le interazioni tra l'utente ed il repository. Tra i possibili si ricorda *TortoiseSVN*, software utilizzato in questo lavoro. Esso è stato sviluppato direttamente dai creatori di SVN. In realtà, TortoiseSVN non fornisce all'utente una vera e propria interfaccia grafica, ma si presenta sotto forma di *shell extension*, ovvero aggiunge ai menu di Windows explorer la voce 'TortoiseSVN' che poi si espande nei vari comandi. Come esempio, questa integrazione è osservabile in figura 9.10.

### Il repository del progetto WQF

Il software di versioning *Subversion* è installato sul server SVRGMEE2 (raggiungibile all'indirizzo `svrgmee2.dei.unipd.it`). Tale server non è accessibile dall'esterno con programmi tipo telnet/ssh, ma solo sulla porta 80 con il protocollo http. Per poter accedere dall'esterno si deve creare un "tunnel". Per evitare questa incombenza, per *SVN* è stata creata una modalità di accesso tra-

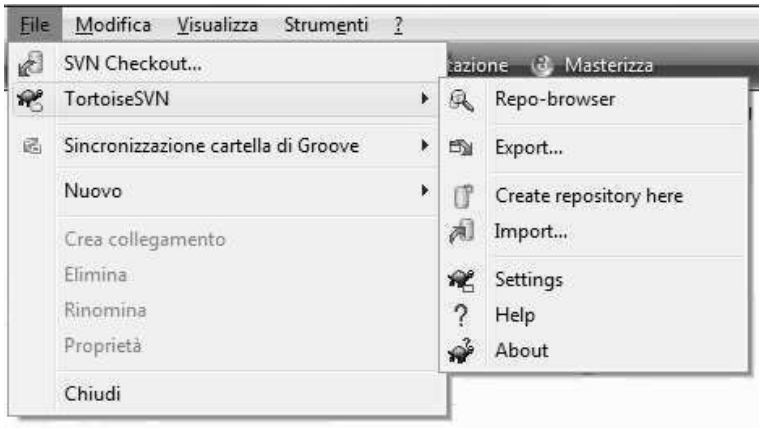


**Figura 9.10.** Shell extension con TortoiseSVN.

mite http. In pratica i *repositories* sono "pubblici" o, meglio, essi sono protetti da password ma, comunque, accessibili da internet.

Il repository per questo lavoro è stato chiamato WQF, dal nome che, provvisoriamente, abbiamo assegnato al progetto. Esso è raggiungibile dal browser web accedendo al seguente indirizzo:

`http://svrgmee2.dei.unipd.it/svnrepos/wqf`

Alla richiesta di accesso il browser risponde chiedendo username e password.

**Albero delle directory** Prima di cominciare il vero e proprio lavoro di sviluppo, è necessario creare l'albero delle directory del repository, in modo tale da poter sempre mantenere organizzati i file ed i documenti prodotti. La struttura delle directory del repository WQF è illustrata nella figura 9.11. Si possono individuare tre principali aree:

- `docs`;
- `source`;
- `tools`.

Nei prossimi paragrafi sarà spiegato cosa ogni cartella andrà a contenere con l'avanzare dello sviluppo del progetto, specificando anche il significato di eventuali sottocartelle.

**La cartella `docs`** Per i nomi delle diverse cartelle dell'albero sono stati utilizzati nomi in inglese. Come facilmente intuibile, la cartella `docs` contiene tutti i documenti prodotti, opportunamente organizzati nelle seguenti sottocartelle:


**Figura 9.11.** Albero delle directory del repository WQF.

- `architecture_design`: è la cartella in cui sono conservati i documenti che raccolgono tutte le ipotesi e gli studi sul progetto dell'architettura dell'applicazione.

- `db`: in questa directory vengono memorizzati i documenti riguardanti il database sul quale "gira" il motore di *workflow* come, ad esempio, il diagramma E-R finale.

- `functional_analysis`: i documenti in questa cartella riguardano l'analisi funzionale dell'intero sistema.

- `interface_design`: qui sono raccolti i progetti delle interfacce grafiche con le quali l'utente finale interagirà con il sistema.

- `manuals`: i manuali del sistema redatti (manuale utente, manuale di istallazione e quello di amministrazione) sono qui conservati.

- `template`: qui vengono depositati tutti i template utilizzati per produrre la documentazione.
- `test_design`: la directory contiene il documento che descrive le azioni da compiere per effettuare il test del sistema.
- `thesis`: contiene la presente tesi e lavori correlati.
- `tools_analysis`: contiene i documenti che descrivono il funzionamento e l'utilizzo dei diversi *tools* utilizzati. Questo documento sul software *Subversion*, ad esempio, è uno di quelli.

**La cartella `source`** La cartella `source` contiene tutto il codice prodotto.

- `apps`: in questa cartella viene conservato il codice relativo alla vera e propria applicazione. Probabilmente saranno presenti delle sottocartelle che variano, però, a seconda del linguaggio e dei tools di sviluppo scelti.
- `db`: questa sezione conserva i sorgenti del database.
- `examples`: contiene i sorgenti delle applicazioni di prova (di dimesioni ridotte) realizzate per prendere confidenza con i tools di sviluppoi.
- `install_pack`: qui vengono memorizzati i sorgenti del pacchetto di installazione creato a prodotto finito.
- `template`: la directory contiene gli eventuali template usati per scrivere il codice.
- `test`: qui è depositato il codice di tutti gli script creati per effettuare il test del prodotto.

**La cartella `tools`** Questa cartella contiene semplicemente i files di installazione di tutti i tools utilizzati nel progetto, nelle versioni scelte nel documento "Analisi tools".

## 9.3 Flusso di una richiesta `HTML`

Una richiesta (*request*) `HTML` è una serie di pacchetti di dati inviati dal browser dell'utente al server Tomcat attraverso la rete. In questa richiesta sono contenuti, tra le altre informazioni, il nome della pagina che si vuole visualizzare e gli eventuali campi inseriti nei form, in forma di stringa o binaria, a seconda del tipo di form dichiarato. Un'immagine esemplificativa del flusso di una richiesta `HTML` è riportata in figura 9.12.
Nel file `web.xml` si definisce il meccanismo che caricherà le istanze dei componenti che gestiscono le richieste e le risposte `HTTP`. Il `DispatcherServlet` cercherà un

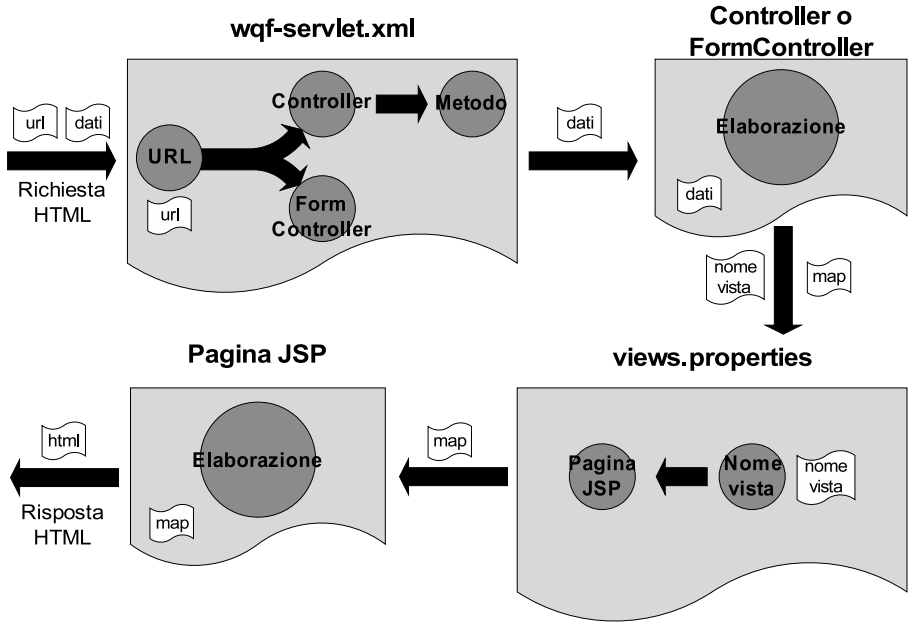


**Figura 9.12.** Esemplificazione del flusso dei dati di una richiesta HTML.

file `WQF-servlet.xml`, inoltre si dichiara che ciascuna URL terminante con `.htm` andrà gestita da tale servlet.

```
<servlet>
    <servlet-name>WQF</servlet-name>
    <servlet-class>org.springframework.web.servlet.
       DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>WQF</servlet-name>
    <url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

**Listing 9.1.** Frammento del file `web.xml`

### 9.3.1 Il ruolo di HandlerMapping e ViewResolver

Con Spring Framework, quando è richiesta una pagina tramite URL, l'interfaccia `handlerMapping` (per essere precisi alcune delle sue varie implementazioni) si occupa di associarvi il relativo `Controller`. Nell'applicazione che si vuole realizzare verranno usati `BeanNameUrlHandlerMapping` e `SimpleUrlHandlerMapping`. Essi sono configurati nel file `WQF-servlet.xml` come nel listato 9.2.

```
1 <bean class="org.springframework.web.servlet.handler.
    BeanNameUrlHandlerMapping">
2     <property name="order" value="1"/>
3 </bean>
4
5 <bean id="urlMapping" class="org.springframework.web.servlet
    .handler.SimpleUrlHandlerMapping">
6     <property name="mappings">
7         <props>
8             <prop key="..."></prop>
9             ...
10            <prop key="/processExecution/executeActivity.htm
                  ">executeActivitySimpleFormController</prop>
11            ...
12            <prop key="..."></prop>
13        </props>
14    </property>
15 </bean>
```

**Listing 9.2.** Frammento del file `WQF-servlet.xml`

Il loro funzionamento è il seguente:

- non appena arriva una richiesta URL il primo `HandlerMapping`, nella fattispecie `SimpleUrlHandlerMapping`, controlla se nella sua lista di proprietà (tag `<props>` alla riga 7) vi è corrispondenza con la richiesta ricevuta, quindi invoca il `Controller` relativo;
- se il primo (priorità `value=0` di default) non riesce a soddisfare la richiesta passa il controllo a `BeanNameUrlHandlerMapping` (priorità `value=1`) il quale cerca nella lista dei **bean** configurati (vedi listato 9.3) un `Controller` con il nome corrispondente all'URL richiesto a cui affidare il controllo. Questo è il caso della richiesta di una pagina per la quale non è stato definito un controller (perchè non necessario). Si noti che quest'ultimo `HandlerMapping` deve essere l'ultimo ad avere il controllo (ciò si ottiene dandogli l'ordine più alto di priorità) altrimenti oscurerebbe il funzionamento dei successivi.

```
1 ...
2 <bean name="/welcome.htm" class="org.springframework.web.
    servlet.mvc.UrlFilenameViewController" />
3 <bean name="/accessDenied.htm" class="org.springframework.
    web.servlet.mvc.UrlFilenameViewController" />
4 <bean name="/login.htm" class="org.springframework.web.
    servlet.mvc.UrlFilenameViewController" />
5 <bean name="/loginError.htm" class="org.springframework.web.
    servlet.mvc.UrlFilenameViewController" />
6 ...
```

**Listing 9.3.** Frammento del file `WQF-servlet.xml`

Nel caso dell'applicazione WQF, sono stati utilizzati tutti controller che implementano l'interfaccia `SimpleFormController` di Spring. In questo tipo di controller viene definita una *command class*, che altro non è che un `JavaBean`. Se, all'interno della richiesta pervenuta dall'utente, uno o più campi del form hanno lo stesso nome dei campi della command class, allora essa avrà quei campi già settati all'interno del `FormController`. Si può, così, utilizzare il `JavaBean` direttamente senza fare "a mano" il parsing della richiesta inoltrata ed il mappaggio sul JavaBean. In ogni caso, nel controller, si possono eseguire tutte le operazioni necessarie a gestire le richieste `HTML`: manipolare i dati immessi, interagire con la base di dati (utilizzando i metodi messi a disposizione dal service layer), cambiare a piacimento la risposta (*response*), che il server invia al browser client, raccogliere e organizzare i dati da passare alla vista successiva.

Per fare il passaggio dei dati dal controller alla `JSP` successiva, si usa una particolare struttura dati `Java`: `java.util.Map`. In essa si può immettere qualsiasi oggetto di tipo `Object` e associarvi un nome in una stringa. In particolare è comodo inserivi `java.util.List` o `JavaBean` provenienti da interrogazioni alla base di dati, ma anche stringhe o booleani per, poi, utilizzarli nella `JSP` tramite appositi tag.

Successivamente, quando il `Controller` richiede di visualizzare una pagina, interviene l'interfaccia `ViewResolver`, tramite una delle sue implementazioni. Nel file `WQF-servlet.xml` (si veda il listato 9.4) sono state utilizzate `UrlBasedViewResolver` e `ResourceBundleViewResolver`.

```
1 <bean id="urlBasedViewResolver" class="org.springframework.
    web.servlet.view.UrlBasedViewResolver">
2     <property name="viewClass" value="org.springframework.
        web.servlet.view.JstlView"/>
3 </bean>
4
5 <bean id="viewResolverUser" class="org.springframework.web.
    servlet.view.ResourceBundleViewResolver">
6     <property name="basename" value="views"/>
7     <property name="order" value="0"/>
8 </bean>
```

**Listing 9.4.** Frammento del file `WQF-servlet.xml`

Le caratteristiche della loro implementazione sono:

- `ResourceBundleViewResolver` cerca il nome logico della vista che deve tornare leggendo il file il cui nome è definito nelle sue proprietà (riga 6), la cui estensione è `.properties` e che deve essere collocato alla radice della directory contenente le classi compilate. Se il nome logico della vista richiesta è stato trovato lo stesso file di configurazione specifica anche il tipo di `view` da utilizzare;

- se il primo non trova corrispondenze nella sua configurazione passa il controllo a `UrlBasedViewResolver` che cercherà una vista del tipo configurato

in riga 2, nella fattispecie una JSP, il cui nome logico corrisponde all'URL di tale vista. Si noti che questo ViewResolver deve essere collocato come ultimo, pena l'oscuramento del funzionamento di quelli dopo di lui. Inoltre, tale implementazione offre la possibilità di agevolare il RedirectPattern riconoscendo il comando redirect automaticamente come una view di redirezione e crea un URL di richiesta che verrà passata a uno dei due ViewResolver.

## 9.4 Aggiungere una nuova JSP

In questa sezione viene presentata una guida che illustra il procedimento di inserimento di una nuova pagina JSP all'interno di un progetto Tomcat.

Si supponga di voler creare una nuova JSP, chiamata newUser.jsp, con la quale inserire un nuovo utente nel database. La pagina conterrà un form per l'inserimento dei dati richiesti. Le operazioni da effettuare sono le seguenti:

- creare la pagina newUser.jsp, all'interno della cartella /WQF/WebContent/WEB-INF/jsp o di una sua sottocartella, ad esempio /WQF/WebContent/WEB-INF/jsp/subDir/;
- creare il controller per la pagina. Per fare ciò è sufficiente realizzare una classe java che implementi l'interfaccia Controller di Spring, o che estenda una classe che già la implementa (ad esempio SimpleFormController. In particolare, nel caso dell'esempio in questione, si deve creare il file NewUserSimpleFormController.java nella cartella /WQF/src/. Il listato 9.5 riporta un esempio dell'implementazione del controller.

```
1 public class NewUserSimpleFormController extends
     SimpleFormController{
2
3     public StartProcessSimpleFormController() {
4     setCommandClass(User.class);
5     setCommandName("userDetails");
6     setFormView("newUserView");
7     setSuccessViewView("newUserSuccessView")
8     }
9 }
```

**Listing 9.5.** Esempio di implementazione del file NewUserSimpleFormController.java

Alle righe 4 e 5 viene definita la classe e il nome dell'oggetto *command*. L'oggetto passato al metodo setCommandClass, nell'esempio User.class, deve rispettare i vincoli dei JavaBean. Alle righe 6 e 7, invece, vengono definiti i nomi logici delle viste associate al controller. In particolare, la FormView è la vista caricata alla richiesta della pagina, mentre la SuccessView è quella richiesta in seguito alla chiamata della funzione submit del form;

- decidere un URL da utilizzare come link nelle JSP per richiedere la pagina. Si può, ad esempio, usare /subDir/newUser.htm. Ciascun link nelle pagine avrà la seguente sintassi:

```
<a href="<c:url value="/subDir/newUser.htm"/>">nuovo
   utente</a>
```

  Si noti che l'indirizzo scelto è all'interno del tag <c:url> fornito dalle librerie JSTL;

- modificare il file WQF-servlet.xml (in /WQF/WebContent/WEB-INF/), aggiungendo la mappatura tra la stringa scelta al punto precedente ed il relativo controller, come illustrato nel listato 9.6:

```
<bean id="urlMapping" class="org.springframework.web.
   servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            ...
            <prop key="/subDir/newUser.htm">
               newUserSimpleFormController</prop>
            ...
        </props>
    </property>
</bean>

...


<bean id="newUserSimpleFormController" class="
   NewUserSimpleFormController"></bean>
```

**Listing 9.6.** Estratto del file WQF-servlet.xml

- aggiungere file view.properties (in /WQF/build/classes/) le seguenti stringhe:

```
newUserView(parent)=parent-view
newUserView.url=/WEB-INF/jsp/subDir/newUser.jsp
...
newUserSuccessView(parent)=parent-view
newUserSuccessView.url=/WEB-INF/jsp/subDir/
   newUserSuccess.jsp
```

  newUserView è il nome logico della vista che viene utilizzato nel controller per richiamarla, mentre /WEB-INF/jsp/subDir/newUser.jsp è l'URL della pagina JSP nell'albero delle directory dell'applicazione. La successView può essere una qualsiasi pagina dell'applicazione, volendo anche la stessa pagina del form.

## 9.5 Introduzione agli Annotated Controllers

Negli ultimi anni si sta diffondendo la tendenza ad utilizzare le *annotazioni* come metodo di configurazione di alcuni tipi di dati, anzichè usare i file XML. Per facilitare lo sviluppo di questo pattern di progettazione, Spring, dalla versione 2.5, fornisce supporto per configurare i componenti del framework MVC attraverso l'annotazione. Dalla versione 3.0 di Spring, invece, la classe `SimpleFormController`, ampiamente utilizzata all'interno del progetto WQF, è stata deprecata.

Per definire un controller non è più necessario estendere una classe `Controller` o far riferimento alle Servlet API, ma è sufficiente aggiungere l'annotazione `@Controller` nell'intestazione della classe. In questo modo, una qualsiasi classe potrà essere utilizzata come controller, ad esempio:

```
1 @Controller
2 public class MyClass{
3 ...
```

Per ulteriori dettagli sull'uso degli annotated controllers si consulti la *Spring Reference Guide* riportata in bibliografia.

# Capitolo 10

# Applicazione di prova: autenticazione e gestione di permessi

Dopo aver configurato correttamente l'ambiente di lavoro, procurandosi tutti i tools richiesti, si è scelto di testare il sistema sviluppando una semplice applicazione di prova in cui utilizzare i meccanismi di autenticazione e autorizzazione forniti da Spring (Acegi) Security, poichè la sicurezza è una componente fondamentale per un'applicazione web ben fatta.

Il nome dato all'applicazione di test è `security_project` ed il suo funzionamento è illustrato in figura 10.1. In pratica, l'utente effettua l'accesso inserendo le proprie credenziali nel form della pagina di login. Nel caso in cui l'autenticazione vada a buon fine, l'utente accede alla pagina dei contenuti, in cui quest'ultimi variano in base ai permessi concessi all'utente. In caso contrario, invece, viene visualizzato un messaggio di errore e viene chiesto all'utilizzatore di reinserire le proprie credenziali di accesso.

## 10.1 Autenticazione utenti

All'applicazione `security_project` possono accedere solamente gli utenti elencati nella tabella 10.1. Quando si vogliono utilizzare le funzionalità per la

| Login | Password | Ruolo | Permesso 1 | Permesso 1 |
|---|---|---|---|---|
| mario | rossi | admin | write | read |
| giuseppe | verdi | user | read | write |
| eva | adamo | progettista | write | null |
| dio | omni | universo | write | write |

**Tabella 10.1.** Utenti dell'applicazione `security_project` e relativi permessi.

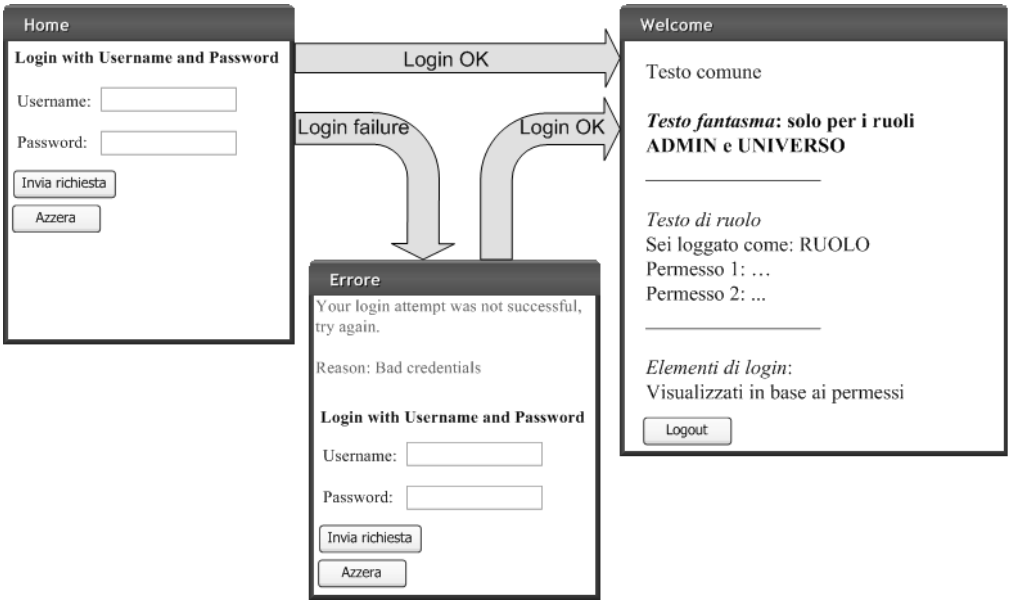


**Figura 10.1.** Rappresentazione grafica del funzionamento dell'applicazione `security_project`.

sicurezza messe a disposizione da Spring framework nella propria applicazione è necessario effettuare le opportune configurazioni. La prima cosa da fare è configurare il file `web.xml` inserendo il seguente frammento di codice:

```
1  <-- Estratto del file web.xml -->
2
3  <filter>
4      <filter-name>springSecurityFilterChain</filter-name>
5      <filter-class>org.springframework.web.filter.
          DelegatingFilterProxy</filter-class>
6  </filter>
7
8  <filter-mapping>
9      <filter-name>springSecurityFilterChain</filter-name>
10   <url-pattern>/*</url-pattern>
11 </filter-mapping>
```

Tutto ciò permette di utilizzare l'infrastruttura web di Spring Security.
`DelegatingFilterProxy` è una classe di Spring Framework che delega il lavoro a un'implementazione del filtro definita come un *bean* nel contesto dell'applicazione. In questo caso il *bean* si chiama `springSecurityFilterChain` ed è parte integrante dell'infrastruttura per gestire la sicurezza web. Una volta aggiunte queste righe al file `web.xml`, è possibile proseguire con la modifica del file di contesto dell'applicazione chiamato `applicationContext-security.xml`. Per abilitare il sistema di sicurezza web è necessario scrivere:

```
1 <-- Estratto del file applicationContext-security.xml -->
2
3 <http auto-config='true'>
4     <intercept-url pattern="/**" access="ROLE_USER,
          ROLE_PROGETTISTA,ROLE_ADMIN,ROLE_UNIVERSO" requires-
          channel="https" />
5     <form-login default-target-url="/welcome.htm" always-use
          -default-target="true" />
6 </http>
```

Il codice indica semplicemente che si vuole mettere in sicurezza tutti gli URL interni all'applicazione permettendo l'accesso solamente agli utenti autenticati con i ruoli elencati alla riga 4. Tutto ciò permette, quindi, di evitare accessi indesiderati effettuati conoscendo gli URL delle pagine protette. É possibile utilizzare più elementi `<intercept-url>` per definire differenti requisiti di accesso per diversi insiemi di URL, tenendo, però, conto del fatto che essi saranno valutati dal sistema nell'ordine in cui compaiono e che verrà, quindi, utilizzato il primo vincolo corrispondente all'URL richiesto. Se, a questo punto, si fa partire l'applicazione è possibile osservare immediatamente che il browser carica una pagina di login in cui si richiede di inserire le credenziali dell'utente, come raffigurato in figura 10.2. La cosa può stupire notevolmente, in quanto nessu-

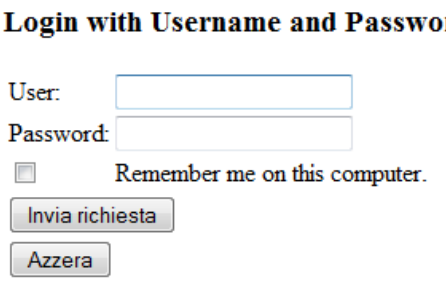


**Figura 10.2.** Pagina di login dell'applicazione `security_project`.

na pagina di login è stata precedentemente creata. La spiegazione è semplice e può anche sorprendere: a meno che non venga esplicitamente impostato l'URL della propria pagina di login, Spring Security si preoccupa di generarne una in automatico. Se, per esempio, si volesse effettuare il login attraverso la pagina `login.jsp` da noi creata, è sufficiente aggiungere il comando

```
1 login-page='/login.htm'
```

all'interno del tag `<form-login>` del listato precedente[1]. Il comando `default-target-url`, indica, invece, al sistema quale pagina caricare dopo aver effettuato

[1]Sarà, inoltre, necessario modificare il file `Security_project-servlet.xml` aggiungendo un tag `<bean>`. Per ulteriori approfondimenti rifarsi alla documentazione di Spring.

l'autenticazione dell'utente. La pagina di login creata in automatico da Spring Security mostra, inoltre, un messaggio di errore nel caso in cui l'autenticazione non vada a buon fine, vedi figura 10.3, come richiesto dai requisiti dell'applicazione. Se, invece, si volesse reindirizzare l'utente che inserisce credenziali non corrette ad una pagina di errore, ad esempio `loginError.jsp`, basta scrivere:

```
1 <form-login login-page="/login.htm" default-target-url="/
    welcome.htm" always-use-default-target="true"
    authentication-failure-url="/loginError.htm" />
```

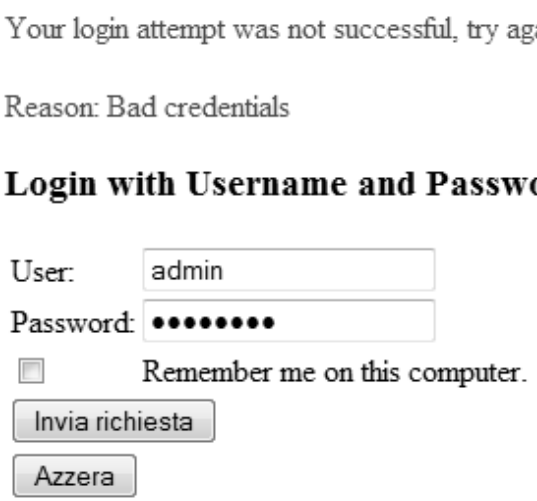


**Figura 10.3.** Errore di autenticazione nell'applicazione `security_project`.

All'interno del tag `<intercept-url>` della riga 4 è stato, inoltre, indicato di utilizzare il protocollo `HTTPS`. Un lettore attento avrà notato che, quando è stato indicato al sistema a quale pagina accedere in seguito ad un'autenticazione effettuata con successo (in questo caso `welcome.jsp`), si è usata l'estensione `.htm` al posto di `.jsp`. Il motivo di tutto ciò consiste nel fatto che si vuole fare in modo che il sistema intercetti un URL che termina con `.htm` e che richiami la pagina associata, con estensione `.jsp` ad esso associata. Le associazioni tra URL e pagine sono indicate nel terzo file di configurazione chiamato `Security_project-servlet.xml` (la convenzione definisce il nome come `nomeapplicazione-servlet.xml`) di cui si riporta un frammento di codice:

```
1 <-- Estratto del file Security_project-servlet.xml -->
2
3 <bean name="/welcome.htm" class="myPack.welcomeController"
    />
4 <bean id="viewResolver" class="org.springframework.web.
    servlet.view.InternalResourceViewResolver">
5     <property name="prefix" value="/WEB-INF/jsp/"/>
```

```
6     <property name="suffix" value=".jsp"/>
7 </bean>
```

Il significato del codice è il seguente: quando si trova un link a `/welcome.htm` si deve richiamare il controller `welcomeController` che reindirizzerà, dopo aver eseguito il proprio lavoro, alla pagina `/WEB-INF/jsp/welcome.jsp`. Se nell'applicazione fossero presenti più pagine, si dovrebbe creare un tag `<bean>` come quello appena visto per ognuna di esse. Il controller `welcomeController` implementato per l'applicazione in esame ha un compito molto semplice, in quanto si limita a generare i permessi dei vari utenti e a passarli alla pagina `welcome.jsp` (come un oggetto `perm` di tipo `Permessi`) che li utilizzerà per filtrare i contenuti da visualizzare. Il codice di `welcomeController` e di seguito riportato:

```
1 <-- Estratto del file welcomeController.java -->
2
3 package myPack;
4
5 import javax.servlet.http.HttpServletRequest;
6 import javax.servlet.http.HttpServletResponse;
7
8 import org.springframework.security.Authentication;
9 import org.springframework.security.context.
     SecurityContextHolder;
10 import org.springframework.web.servlet.ModelAndView;
11 import org.springframework.web.servlet.mvc.Controller;
12
13 public class welcomeController implements  Controller {
14
15     public ModelAndView handleRequest(HttpServletRequest
          request, HttpServletResponse response) throws
          Exception {
16         Authentication auth = SecurityContextHolder.
              getContext().getAuthentication();
17         Permessi perm = new Permessi();
18         if (auth.getName().equals("mario"))
19         {
20             perm = new Permessi(2,1);
21         }
22         if (auth.getName().equals("giuseppe"))
23         {
24             perm = new Permessi(1,2);
25         }
26         if (auth.getName().equals("eva"))
27         {
28             perm = new Permessi(2,0);
29         }
30         if (auth.getName().equals("dio"))
31         {
32             perm = new Permessi(2,2);
```

```
33          }
34          return new ModelAndView("welcome","miopermesso",perm
                );
35      }
36  }
```

Il controller `welcomeController` viene interpellato ogniqualvolta un utente si autentichi dalla pagina di login ed effettui, quindi, una richiesta della pagina `welcome.jsp`. Il controller si occupa della generazione dei permessi in base alle credenziali inserite, i quali vengono passati come parametro alla pagina `welcome` come un oggetto di tipo `Permessi`. Il flusso dei dati verrà, comunque, illustrato nei prossimi paragrafi.

Utilizzando Spring Security non è complesso recuperare le informazioni di autenticazione. Vengono, infatti, messe a disposizione un paio di implementazioni di base utili, che andremo a descrivere nelle prossime righe.

### 10.1.1 In-memory Authentication

Quando si inizia ad integrare Spring Security nella propria applicazione, spesso si desidera, per semplicità, evitare di dover prelevare le informazioni per l'autenticazione degli utenti da una fonte di persistenza (ad esempio una base di dati) sebbene, come vedremo in seguito, tutto ciò è tutt'altro che complicato. Per configurare l'applicazione in queste situazioni, non si deve fare altro che aggiungere il seguente frammento di codice nel file `applicationContext-security.xml` che viene caricato dal sistema nel momento in cui viene letto il file `web.xml` all'avvio dell'applicazione.

```
1  <-- Estratto del file applicationContext-security.xml -->
2
3  <!-- **********IN-MEMORY AUTHENTICATION********** -->
4
5  <authentication-provider>
6      <user-service>
7          <user name="mario" password="rossi" authorities="
               ROLE_ADMIN, ROLE_USER" />
8          <user   name="giuseppe" password="verdi" authorities
               ="ROLE_USER" />
9          <user name="eva" password="adamo"   authorities="
               ROLE_PROGETTISTA" />
10         <user name="dio" password="omni" authorities="
               ROLE_UNIVERSO" />
11         <user name="sadmin" password="sadmin" authorities="
               ROLE_USER,ROLE_UNIVERSO,ROLE_PROGETTISTA,
               ROLE_ADMIN" />
12     </user-service>
13 </authentication-provider>
```

### 10.1.2 JDBC Authentication

Spring Security offre anche la possibilità di leggere le informazioni per l'autenticazione degli utenti da una sorgente dati JDBC. I driver JDBC sono usati internamente a Spring e questo risparmia all'utente la fatica di dover realizzare un ORM (Object Relational Mapper) solamente per memorizzare i dettagli degli utenti. Le righe seguenti rappresentano la configurazione che serve a fare tutto ciò per l'applicazione `security_project` sempre nel file `applicationContext-security.xml`. Ovviamente il frammento di codice che riguarda l'autenticazione "in-memory" deve essere rimosso o commentato.

```
<-- Estratto del file applicationContext-security.xml -->

<!-- ************JDBC AUTHENTICATION************ -->

<authentication-provider user-service-ref="
   daoAuthenticationProvider">
</authentication-provider>

<beans:bean id="daoAuthenticationProvider" class="org.
   springframework.security.userdetails.jdbc.JdbcDaoImpl">
    <beans:property name="dataSource" ref="dataSource" />
</beans:bean>

<!-- Spring managed webcosts datasource -->
<beans:bean id="dataSource" class="org.springframework.jdbc.
   datasource.DriverManagerDataSource">
    <beans:property name="driverClassName" value="org.
       postgresql.Driver" />
    <beans:property name="url" value="jdbc:postgresql
       ://127.0.0.1:5432/sec_proj" />
    <beans:property name="username" value="postgres" />
    <beans:property name="password" value="admin" />
</beans:bean>
```

Non è necessario specificare alcun riferimento alla tabella del database nel caso in cui si utilizzi lo schema di default suggerito da Spring Security e di seguito riportato.

```
CREATE TABLE users (
username VARCHAR(50) NOT NULL PRIMARY KEY,
password VARCHAR(50) NOT NULL,
enabled BIT NOT NULL
);

CREATE TABLE authorities (
username VARCHAR(50) NOT NULL,
authority VARCHAR(50) NOT NULL
);

```

```
12 ALTER TABLE authorities ADD CONSTRAINT fk_authorities_users
       foreign key (username) REFERENCES users(username);
```

La tabella `users` contiene le informazioni riguardanti gli utenti, ovvero username, password e stato dell'account (*enabled* o *disabled*). La tabella `authorities` fornisce, invece, la lista dei ruoli di ciascun utente (un utente con più ruoli vedrà il suo username presente su più record).

È, comunque, possibile definire schemi personalizzati nella base di dati. È sufficiente, poi, indicare al sistema dove prelevare i dati degli utenti come fa il seguente frammento di codice di esempio:

```
1  <beans:bean id="daoAuthenticationProvider" class="org.
       springframework.security.userdetails.jdbc.JdbcDaoImpl">
2      <beans:property name="dataSource" ref="dataSource"/>
3      <beans:property name="usersByUsernameQuery">
4          <beans:value>
5              SELECT USERNAME, PASSWORD, ENABLED FROM USERS
                   WHERE USERNAME=?
6          </beans:value>
7      </beans:property>
8      <beans:property name="authoritiesByUsernameQuery">
9          <beans:value>
10             SELECT uat.USERNAME AS USERNAME, AUTHORITY FROM
                   AUTHORITIES uat, USERS ut WHERE uat.USERNAME
                    = ut.USERNAME AND uat.USERNAME=?
11         </beans:value>
12     </beans:property>
13 </beans:bean>
```

Il meccanismo usato per prelevare le informazioni è adattabile, quindi, a qualsiasi base di dati che disponga delle informazioni precedentemente elencate. La cosa fondamentale è modificare le query `usersByUsernameQuery` e `authoritiesBy UsernameQuery` adattandole al proprio database in modo che esse restituiscano rispettivamente `username-password-stato` e `username-ruolo`.

## 10.2 Autorizzazione sui ruoli

Ritornando al funzionamento dell'applicazione precedentemente introdotto, si è detto che vi sono degli elementi della pagina dei contenuti che vengono visualizzati in base al ruolo dell'utente che si è autenticato. La figura 10.4 mostra la differenza tra quello che vede un utente autenticato con ruolo ROLE_ADMIN, nella fattispecie l'utente "mario", ed uno con ruolo ROLE_PROGETTISTA, l'utente "eva". Si nota che l'utente con ruolo ROLE_ADMIN visualizza un contenuto che l'altro non vede, ovvero la scritta in grassetto. Per gestire le autorizzazioni sui contenuti, Spring Security mette a disposizione il tag `<authz:authorize>` che va utilizzato direttamente nelle pagine dell'applicazione. In particolare, nella

pagina `welcome.jsp` dell'applicazione `security_project`, ovvero la pagina dei contenuti, è presente il seguente frammento di codice:

```
<authz:authorize ifAnyGranted="ROLE_ADMIN,ROLE_UNIVERSO" >
    <h3>
        scritta riservata a chi ha permesso universo o admin
    </h3>
</authz:authorize>
```

Nel tag è presente l'attributo `ifAnyGranted` seguito da una lista di ruoli. È sufficiente che l'utente appartenga ad uno dei ruoli elencati. Sono, inoltre, disponibili gli attributi `ifAllGranted` e `ifNotGranted` i cui significati sono facilmente intuibili. Il frammento di pagina racchiuso da questo tag viene, quindi, visualizzato solo se i requisiti di autorizzazione sono soddisfatti.

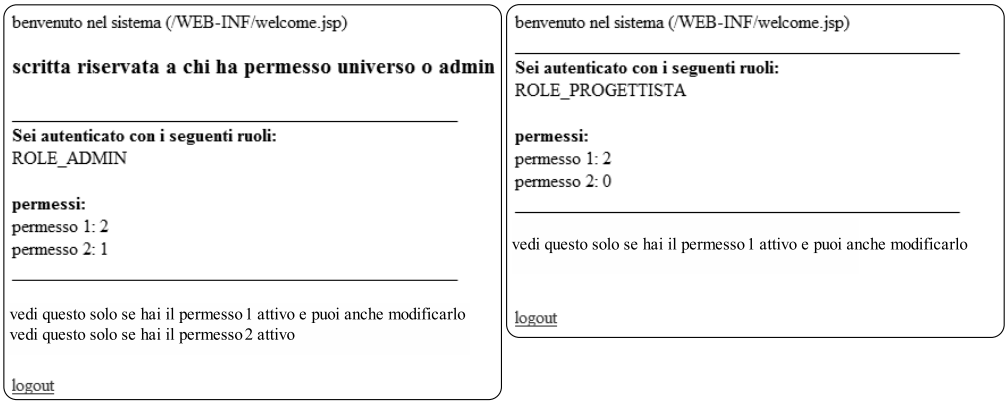


**Figura 10.4.** I contenuti visualizzati dall'utente "mario" (a sinistra) e quelli visti dall'utente "eva" (a destra).

## 10.3 Autorizzazione sui singoli permessi

Oltre alle autorizzazioni in base ai ruoli dell'utente autenticato, i contenuti dell'applicazione dovevano essere "filtrati" anche in base ai permessi specifici assegnati ai singoli utenti. Per semplicità i permessi sono stati indicati con dei numeri interi dove 0 corrisponde a "nessun permesso", 1 a "lettura" e 2 a "scrittura". Dopo aver effettuato l'autenticazione, Spring Security passa il controllo al controller `welcomeController.java`, il quale restituisce un oggetto `perm` della classe `Permessi.java` indicante i permessi assegnati all'utente che ha effettuato il login e richiama la pagina `welcome.jsp`. È, inoltre, possibile restituire più oggetti alla pagina utilizzando l'interfaccia `java.util.map`[2]. Per differenziare

[2]Vedi documentazione `Java`

i contenuti in base ai permessi assegnati, sono stati creati due tag personalizzati[3], chiamati rispettivamente `permesso1.tag` e `permesso2.tag`, nella cartella `/WEB-INF/tags` dei quali riportiamo il codice.

```
<!-- permesso1.tag -->
<%@attribute name="perm_value" required="true "%>
<%
    if (Integer.parseInt(perm_value) > 0)
    {
        out.print("vedi questo solo se hai il permesso 1
           attivo");
    }
    if (Integer.parseInt(perm_value) == 2)
    {
        out.print("  e puoi anche modificarlo");
    }
%>
```

```
<!-- permesso2.tag -->
<%@attribute name="perm_value" required="true "%>
<%
    if (Integer.parseInt(perm_value) > 0)
    {
        out.print("vedi questo solo se hai il permesso 2
           attivo");
    }
    if (Integer.parseInt(perm_value) == 2)
    {
        out.print("  e puoi anche modificarlo");
    }
%>
```

Per utilizzare i tag nelle pagine dell'applicazione (per esempio in `welcome.jsp` è sufficiente scrivere:

```
<tags:permesso1 perm_value="${miopermesso.p1}"/>
<tags:permesso2 perm_value="${miopermesso.p2}"/>
```

dove `perm.p1` e `perm.p2` corrispondono ai valori dei permessi dell'utente autenticato restituiti da `welcomeController`. Si osservi che la cartella `/WEB-INF/tags` è riconosciuta in automatico dal sistema che, quindi, sa dove andare a leggere il contenuto dei tag personalizzati inseriti.

Tornando all'esempio visualizzato in figura 10.4, è possibile osservare come il contenuto del tag `permesso2` non venga visualizzato dall'utente "eva" che non dispone di alcun permesso di tipo 2.

[3]In passato risultava abbastanza complicato scrivere tag `JSP` personalizzati. Con l'introduzione dei *tag files* nelle `JSP 2.0` le cose si sono notevolmente semplificate.

# Capitolo 11

# Definizione della base di dati

In questo capitolo si descrive la struttura della base di dati utilizzata dall'applicazione WQF il cui schema fisico è rappresentato in figura 11.1 (pag. 153). Si noti che le tabelle sono state raggruppate tramite differenti colorazioni ciascuna corrispondente a diverse funzionalità dell'applicazione.

- Le tabelle di colore rosa/arancio contengono *dati statici*, ovvero le informazioni riguardanti il disegno dei processi, i ruoli aziendali ed i diritti mediante i quali è possibile svolgere le attività.

- Le tabelle in verde sono adibite a raccogliere le informazioni sui documenti in input/output alle attività dei singoli processi.

- Le tabelle viola raccolgono i *dati dinamici*, ovvero quelli prodotti dal sistema di workflow in *runtime*, durante l'esecuzione delle diverse istanze di processo. In questa categoria di dati rientrano le informazioni sulle istanze di processi e attività, sugli assegnamenti delle risorse lavorative, sui documenti prodotti e sul cammino percorso all'interno del disegno di ogni singolo processo.

- Le tabelle in grigio chiaro, infine, contengono i dati utilizzati dal sistema per gestire l'autenticazione degli utenti che vi accedono.

La figura 11.2 (pag. 154), invece, mostra la tabella preposta a salvare i *filtri di ricerca* creati dagli utenti per effettuare le ricerche sui dati storici conservati nella base di dati.

Nelle sezioni seguenti verranno approfonditi i significati e le funzioni delle tabelle di ognuna delle parti appena elencate. Per non appesantire la descrizione della base di dati con dettagli troppo minuziosi, si è volutamente scelto di non descrivere le tabelle i cui attributi sono stati chiamati con nomi per quanto possibile esplicativi.

## 11.1 Tabelle per il disegno dei processi

Come accennato in precedenza, le tabelle per il disegno dei processi costituiscono la parte di dati qui definita statica *statica*, in quanto quest'ultimi vengono modificati solamente nella fase di disegno dei processi da parte del process designer e non vengono alterati dall'esecuzione delle varie istanze. Le tabelle `baseline`, `baseline_status`, `process`, `activity` e `activity_type` contengono gli attributi che descrivono, rispettivamente, baseline, processi e relative attività (si veda l'analisi funzionale). La tabella `activity_relation` raccoglie le informazioni sulle connessioni tra le varie attività nel diagramma di flusso del processo. `policy` raccoglie la lista delle politiche di assegnazione delle risorse lavorative rese disponibili dall'applicazione, mentre la tabella `gao` indica quali attività "ordinarie" vengono assegnate dall'esecuzione di una particolare attività di tipo GAO[1]. La tabella `role` contiene l'elenco dei ruoli aziendali che compongono l'organigramma, mentre `login_role` la mappatura tra questi e gli utenti del sistema. Le tabelle `process_acl` ed `activity_acl` raccolgono le informazioni sui permessi assegnati a ciascun ruolo per processi e attività.

## 11.2 Tabelle per la gestione dei documenti

La sezione della base di dati che gestisce i documenti usati o prodotti dalle attività, è costituita dalle tabelle `activity_data` in cui sono si mappano i documenti in ingresso/uscita, `data_type` che definisce i tipi di documento utilizzabili (ad esempio documento di input statico o upload con template), `data_template` in cui si salvano i riferimenti agli URL dei template ed, infine, `static_reference_data` che memorizza gli URL dei documenti statici disponibili.

## 11.3 Tabelle per la raccolta dei dati runtime

I dati contenuti in questa sezione del database sono quelli generati in *runtime* dall'esecuzione del motore di workflow che sta alla base dell'applicazione. La tabella `process_trace` memorizza le istanze di processo create; i campi `level`, `parent_process_trace_id` e `parent_activity_trace_id` servono a gestire i sottoprocessi permettendo al sistema di ritornare all'esecuzione del processo chiamante una volta che il sottoprocesso è stato concluso. La tabella `activity_trace` raccoglie i dati relative alle singole istanze di attività aperte. Un'istanza di attività viene aperta quando la precedente viene chiusa, ma essa non può essere eseguita fino a che tutte le attività che la precedono non sono concluse. Proprio per effettuare questa verifica è stato predisposto il

[1] Si ricorda che GAO è l'acronimo di Group Activity Owner. Le attività di questo tipo comportano assegnazioni di risorse esplicite per un gruppo di attività. Vedi analisi funzionale al capitolo 8.

campo `entered_connector` che memorizza il numero di connettori in ingresso percorsi. Il sistema si occuperà di confrontare il valore del campo con il numero totale di connettori in entrata ed, eventualmente, di rendere disponibile all'esecuzione l'attività ponendo a `true` il valore del campo `open`. Le tabelle `process_assignement` ed `activity_assignement` contengono i record che documentano le persone (login) a cui i processi e le attività che li compongono sono stati assegnati. Quando una nuova istanza di processo viene avviata, viene generato un record nella tabella `unassigned_activity` per ognuna delle attività che lo compongono per indicare che esse non sono state, appunto, assegnate a nessuna persona. Una volta assegnata un'attività, il record corrispondente viene eliminato. La tabella può, quindi, risultare anche completamente vuota. Quando viene aperta un'istanza di attività con policy di assegnamento "Owner escalation", viene aggiunta una tupla alla tabella `runtime_assignement` che tiene, quindi, traccia delle escalation non ancora gestite; una volta effettuato l'assegnamento, la tupla in questione viene eliminata. La tabella `flow_trace` tiene traccia delle connessioni tra attività percorse durante il flusso di lavoro all'interno di un processo e risulta utile per capire quale cammino è stato eseguito per raggiungere una particolare attività nel caso essa sia raggiungibile da più vie. La tabella `log_entry`, infine, serve a memorizzare le informazioni (URL, creatore, data, ecc.) riguardanti i documenti prodotti dall'esecuzione delle attività (upload).

## 11.4 Tabelle per l'autenticazione degli utenti

Questa parte del database contiene le tabelle `login`, `authority` e `authorities` che identificano rispettivamente gli utenti dell'applicazione, i ruoli di sistema (differenti da i ruoli dell'organigramma aziendale) e la mappatura tra di essi. Le tre tabelle formano una struttura simile a quella suggerita da Spring Security illustrata nel capitolo 10.

## 11.5 Tabella per la memorizzazione dei filtri di ricerca

La tabella `search_filter` memorizza i filtri di ricerca creati dai singoli utenti per uso personale o dall'amministratore di sistema per renderli disponibili a gruppi di utenti (per esempio ad un particolare ruolo). Ogni singolo filtro è identificato, oltre che dall'`id` presente in tutte le tabelle, da un nome. I campi rimanenti sono quelli che determinano i criteri di ricerca che può essere suddivisa in tre categorie principali:

- ricerca per processo/attività;
- ricerca di documenti;
- ricerca per ruolo/persona.

Per permettere all'applicazione di appoggiarsi ad una sola tabella della base di dati, utilizzando, quindi, un unico filtro per le tre tipologie di analisi dei dati, è stato necessario creare un attributo `date_reference` che viene usato per eliminare l'ambiguità che producono i campi in cui vengono inserite le date per limitare la ricerca da un punto di vista temporale. Esse, infatti, si possono riferire alla data di apertura/chiusura di un'istanza di processo/attività o alla data di creazione di un particolare documento.

I campi booleani che terminano con i suffissi `_sm` e `_sr` (abbreviazioni di "show mask" e "show result") indicano, invece, quali campi del filtro devono essere mostrati nella maschera di ricerca e quali nella visualizzazione dei risultati prodotti.

Si noti che la tabella `search_filter` contiene le chiavi esterne `creator_login`, `executor_login` e `executor_role` che identificano creatore ed utilizzatori del filtro e che sono collegati rispettivamente alle tabelle `login` e `role` dello schema di fig. 11.1.

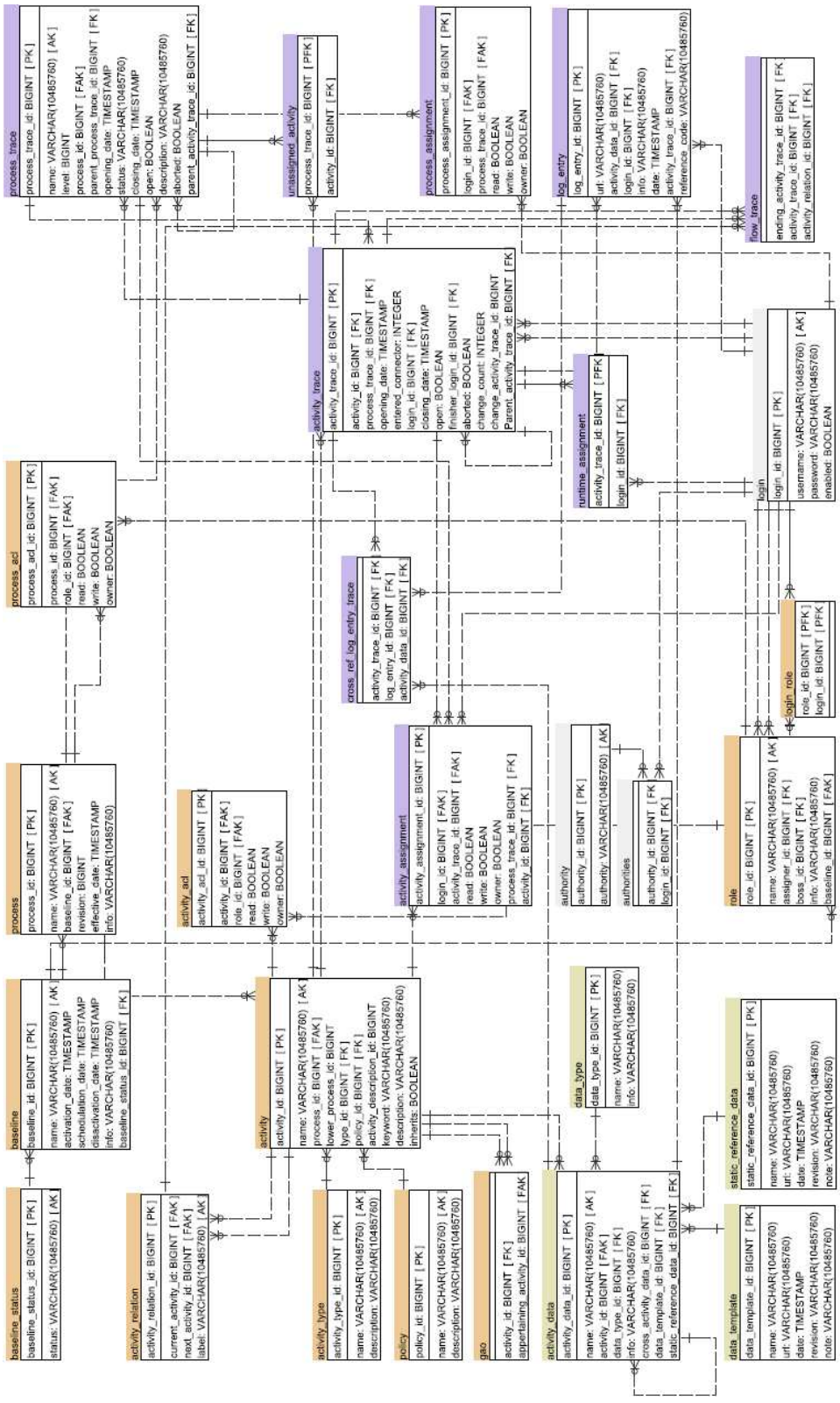


**Figura 11.1.** Schema fisico della base di dati utilizzata dall'applicazione WQF.

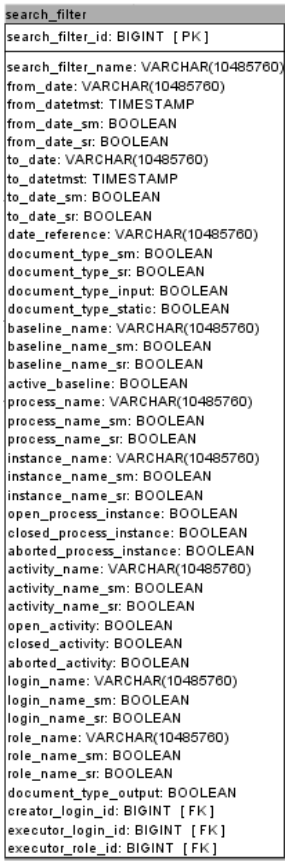


**Figura 11.2.** Tabella per la memorizzazione dei filtri di ricerca sui dati dell'applicazione WQF.

# Capitolo 12

# Manuale dell'installatore

## 12.1 Installazione della Java Virtual Machine

Tomcat, come tutte le applicazioni *Java-based*, richiede una *Java Virtual Machine* (JVM) per funzionare. La Sun Microsystems distribuisce gratuitamente una JVM per Windows, Linux e Solaris. Vendors di terze parti e gruppi open-source distribuiscono JVM per altre piattaforme, non sempre gratuitamente. Prima di installare Tomcat 6 è necessario assicurarsi che la Java 5 JVM sia installata nel proprio sistema. Per verificare quale versione della JVM si ha, è sufficiente utilizzare il comando `java -version` come mostrato.

```
1 C:\> java -version
2 java version "1.6.0_16"
3 Java(TM) SE Runtime Environment, Standard Edition (build
     1.6.0_16-b01)
4 Java HotSpot(TM) Client VM (build 14.2-b01, mixed mode,
     sharing)
```

Una stringa di tipo `1.5.x`, o superiore, indica che si possiede una versione della JVM che permette di installare Tomcat correttamente. Il comando appena descritto presuppone che il percorso `JAVA_HOME/bin` sia contenuto nella variabile di sistema `PATH`. Le sezioni seguenti illustrano il procedimento per l'installazione della JVM in ambiente Windows e Linux. Nel caso si disponga già di una versione della JVM corretta è possibile passare direttamente alla sezione 12.2.

Le versioni di Tomcat precedenti alla `5.5` richiedevano l'installazione del *Java Development Kit* (JDK) e non solo del *Java Runtime Environment* (JRE). Il compilatore `Java` in esso contenuto era utilizzato dalle versioni precedenti di Tomcat per compilare le `JSP` in fase di *runtime* e, quindi, l'installazione del solo JRE non era sufficiente. Le versioni `5.5` e `6` di Tomcat includono un compilatore `Java` (*Eclipse JDT Java compiler*). Questo è usato per compilare le pagine `JSP` e, quindi, è possibile eseguire Tomcat `5.5` e `6`, solo con il JRE.

### 12.1.1 Installazione della JVM in ambiente Windows

Per prima cosa si deve scaricare l'ultima versione di JDK o JRE dal sito Web `http://java.sun.com`. A questo punto è sufficiente far partire l'eseguibile scaricato e, dopo aver seguito la semplice procedura di installazione, si avrà JVM/-JRE installato nel proprio sistema. La cartella in cui sono stati installati i files è nota come *Java Home*. Essa si ramifica in numerose sottocartelle, ma quella di maggior interesse è `bin`, nella quale sono contnuti i vari eseguibili. Il passo successivo consiste nell'aggiungere la cartella *Java Home* in una variabile di sistema chiamata `JAVA_HOME`, in modo tale che essa possa essere trovata quando viene invocata. La sottocartella `bin` deve, invece, essere aggiunta alla variabile `PATH`. Per ottenere informazioni sul procedimento di creazione di una variabile di sistema si consulti il sito `http://support.microsoft.com`.

### 12.1.2 Installazione della JVM in ambiente Linux

Le versioni di JDK/JRE per Linux sono, a loro volta, disponibili, all'indirizzo `http://java.sun.com`. Una buona alternativa alla JVM offerta da Sun per Linux è l'implementazione di IBM[1]. Le istruzioni a seguire si riferiscono, comunque, alla distribuzione della Sun. La piattaforma ufficialmente supportata è Red Hat Linux, ma i pacchetti JDK e JRE della Sun funzionano senza apparenti problemi su tutte le altre distribuzioni.

Se si è scaricato un archivio tar/gzip, la procedura di installazione è la seguente. Per prima cosa, una volta scaricato il file si deve estrarne il contenuto. Per installare JDK per tutti gli utenti è necessario eseguire i comandi che seguono dopo essersi autenticati come `root` o utilizzando il comando `sudo`. A questo punto si deve spostare il file precedentemente estratto nella cartella in cui lo si vuole installare. Se si sta installando JDK (o JRE) per uno specifico utente, lo si deve installare nella cartella *Home* dell'utente stesso. In alternativa, la cartella di default è `/usr/java/jdk-[version number]`, dove `[version number]` corrisponde alla versione di JDK che si sta installando. Aggiungere, ora, i permessi di esecuzione al file scrivendo

```
1 # chmod o+x jdk-1_6_0_16-linux-i586.bin
```

ed eseguire il file con il comando

```
1 # ./jdk-1_6_0_16-linux-i586.bin
```

Prima di iniziare l'installazione, viene visualizzato il contratto di licenza che si deve accettare. Ad installazione completata, invece, si deve aggiungere al sistema la variabile `JAVA_HOME` con la posizione del pacchetto nel file system. Questo valore deve essere aggiunto al file `~/.bashrc` per la propria utenza, o a `/etc/profile` per tutti gli utenti.

[1] http://www.ibm.com/developerworks/java/jdk/

```
JAVA_HOME=/usr/java/jdk-1_6_0_16-linux-i586/
export JAVA_HOME
PATH=$JAVA_HOME/bin:$PATH
export PATH
```

Per testare l'installazione è sufficiente digitare `javac` nella shell. Questo dovrebbe produrre il seguente output (ritagliato per ragioni di brevità):

```
Usage: javac <options> <source files>
where possible options include:
-g                         Generate all debugging info
-g:none                    Generate no debugging info
-g:{lines,vars,source}     Generate only some debugging info
```

Per installare JVM usando un *installer* RPM, invece, si deve per prima cosa, scaricare il file nel formato `j2sdk-[version number]-linux-i586-rpm.bin`. Eseguendo questo file viene presentato il contratto di licenza per Apache e, al termine dell'esecuzione, viene creato un file RPM con lo stesso nome, ma senza il suffisso `.bin`. Per eseguire il file RPM ci si deve loggare come `root` o utilizzare il comando `sudo`, aggiungere i permessi di esecuzione e, quindi, eseseguire il file:

```
# chmod o+x jdk-1_6_0_16-linux-i586-rpm.bin
# ./jdk-1_6_0_16-linux-i586-rpm.bin
```

Per permettere che l'installazione prosegua è necessario accettare i termini di licenza visualizzati. Una volta accettato il contratto di licenza, lo script di installazione andrà a creare il file RPM nella directory corrente. Per installare l'RPM scrivere:

```
# rpm -iv jdk-1_6_0_16-linux-i586-rpm.bin
```

Con questa procedura si è installato JDK in `/usr/java/jdk1.6.0_16`.

## 12.2 Installazione di Tomcat

Nel proseguo del testo si indicherà la directory di installazione di Tomcat con `TOMCAT_HOME` o `CATALINA_HOME`. Le varie distribuzioni dei files di installazione di Tomcat per le diverse piattaforme possono essere reperite all'indirizzo web `http://tomcat.apache.org`. Il sito web di Tomcat elenca quattro differenti distribuzioni per ciascuna versione di Tomcat: *core*, *deployer*, *embedded* e *admin webapp*. Un utente che si avvicina per la prima volta a Tomcat può trovarsi, quindi, in serio imbarazzo nello scegliere la versione da scaricare. Le distribuzioni sono:

- Core: è la versione base do Tomcat ed è disponibile come file `tar.gz`, `ZIP` o come eseguibile per Windows (`.exe`).
- Deployer: si tratta del deployer per l'applicazione web Tomcat *standalone*.

- Admin Webapp: attualmente questa distribuzione non è disponibile per Tomcat `6.x`. Nelle versioni precedenti si trattava di un'applicazione web che veniva utilizzata per amministrare Tomcat e che si trovava in un package differente per motivi di sicurezza.

Per lo sviluppo di un'applicazione Web da caricare su un server sul quale è installato Tomcat, come nel caso dell'applicazione WQF, si deve scaricare la distribuzione *core*.

### 12.2.1 Installazione di Tomcat in ambiente Windows

La pagina di download presenta differenti link per Tomcat 6.x. Quella desiderata dagli utenti Windows presenta estensione `.exe`. Dopo aver salvato il file nella cartella desiderata, è sufficiente eseguire il file per far partire l'installazione. Come spesso accade, l'installer chiede di accettare la licenza Apache, dopodichè viene presentata una schermata in cui si chiede di scegliere i componenti da installare (vedi figura 12.1). É conveniente optare per un'installazione completa, con la

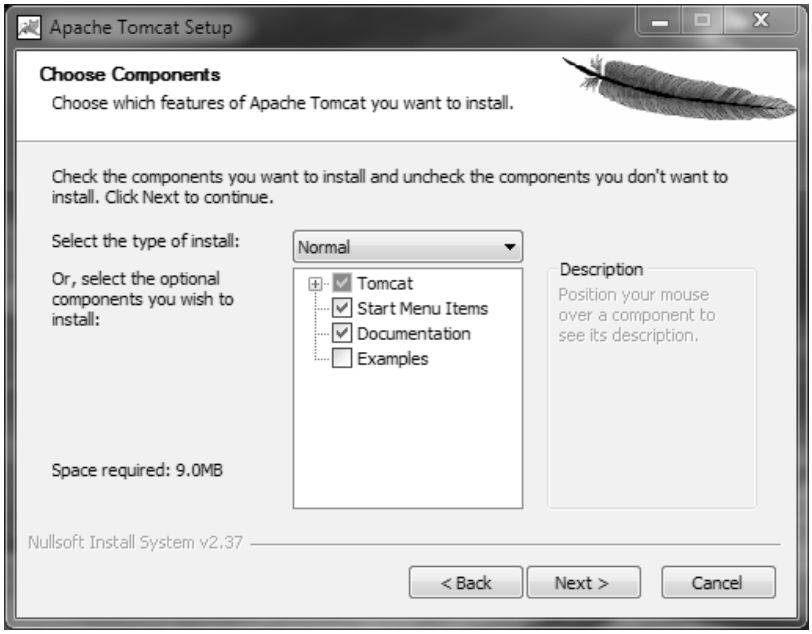


**Figura 12.1.** Scelta dei componenti da installare durante l'installazione di Tomcat.

quali verranno installati tutti i componenti di Tomcat. Alcuni di essi meritano una breve descrizione. Uno dei componenti che non verrebbe installato di default, ma che risulta essere molto utile, è il componente *Service* (selezionabile espandendo il sottomenu "Tomcat"). Questo componente, una volta installato, permette di avviare, terminare o far ripartire Tomcat come un qualsiasi servizio di Windows. Uno dei vantaggi offerti da questa opzione è, quindi, la possibilità di far avviare

Tomcat assieme al sistema operativo. L'nstallazione prosegue chiedendo di indicare la directory di installazione, la password di amministrazione che si desidera e la directory in cui si trova JDK (spesso viene rilevata in automatico). Al termine dell'installazione, anche se non è obbligatorio, è consigliabile aggiungere ad una nuova variabile di sistema chiamata `CATALINA_HOME` il path di installazione. Prima di dichiarare concluso il processo di installazione, è necessario verificare che tutto sia stato installato correttamente. Per fare tutto ciò, dopo aver avviato Tomcat (manualmente eseguendo `CATALINA_HOME/bin/tomcat6.exe` o, automaticamente, impostandolo come servizio di Windows), è sufficiente aprire un browser e accedere all'indirizzo `http://localhost:8080` e verificare che appaia una pagina come quella di figura 12.2.

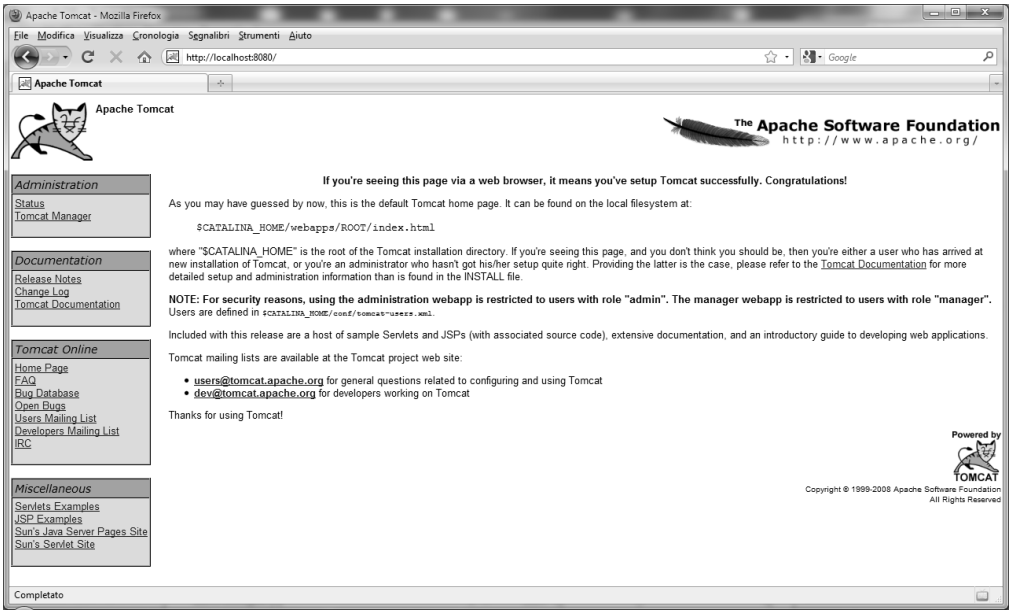


**Figura 12.2.** L'home page di Tomcat.

### 12.2.2 Installazione di Tomcat in ambiente Linux

L'installazione di Tomcat in ambiente Linux è molto semplice. Dopo aver scaricato il file `tar/gzip` dal sito di Tomcat, è sufficiente decomprimerlo nella cartella in cui lo si desidera installare (ad esempio `/usr/java/tomcat-6`. A questo punto si deve esportare la variabile `$CATALINA_HOME` (in `bash`) utilizzando i seguenti comandi:

```
1 # CATALINA_HOME=/usr/java/tomcat-6
2 # export CATALINA_HOME
```

A questo punto è possibile avviare Tomcat scrivendo:

```
# CATALINA_HOME/bin/startup.sh
```

Un altro possibile modo per installare Tomcat è usare gli strumenti per la gestione dei pacchetti specifici delle diverse distribuzioni di Linux. Questo risulta comodo per molti utenti, ma va sottolineato che questi tools spesso posizionano i files di configurazione in locazioni non standard. É possibile verificare il buon esito dell'installazione con lo stesso metodo usato in ambiente Windows.

## 12.3 Configurazione dell'IDE Eclipse

Eclipse, essendo scritto in linguaggio Java, non richiede una vera e propria installazione. É sufficiente, infatti, scaricare l'archivio `Eclipse IDE for Java EE Developers` all'indirizzo `http://www.eclipse.org/download` (accertarsi di scaricare la versione *Galileo*), scompattarlo ed eseguire il file `eclipse.exe`. É possibile copiare la cartella di Eclipse da una macchina all'altra mantenendo la sua operativitá.

É disponibile un plugin per Eclipse che interfaccia l'IDE con Tomcat denominato *Sysdeo Eclipse Tomcat Launcher plugin* e reperibile all'indirizzo `http://www.eclipsetotale.com/tomcatPlugin.html`. Per installarlo è sufficiente decomprimere l'archivio nella cartella `plugins` di Eclipse. Tale plugin permette di:

- inizializzare e fermare Tomcat 4.x, 5.x e 6.x;
- registrare i processi Tomcat nel debugger di Eclipse;
- creare progetti WAR (modificando il file server.xml mediante l'uso di un wizard);
- aggiungere progetti Java al classpath di Tomcat;
- settare i parametri JVM di Tomcat, i classpath e i bootclasspath;
- esportare progetti Tomcat come file WAR.

Una volta installato correttamente il plugin, è dunque possibile realizzare progetti con la dicitura "Is a Tomcat Project", aggiungendo in automatico tutte le librerie necessarie al funzionamento Tomcat. Le librerie di Spring vanno, invece, aggiunte manualmente.

## 12.4 Attivazione del canale HTTPS

Per ottenere informazioni su come abilitare l'utilizzo di un canale di sicurezza tramite `HTTPS` per le applicazioni web che usano Tomcat, è possibile consultare la guida all'indirizzo `http://localhost:8080/docs/ssl-howto.html`. A titolo

di esempio si riporta la configurazione dell'ambiente di lavoro in cui è stato sviluppato il progetto WQF. Da terminale eseguire:

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
Immettere la password del keystore: password
Specificare nome e cognome:
 [Unknown]: Enrico Righetto
Specificare il nome dell'unità aziendale:
 [Unknown]: DEI
Specificare il nome dell'azienda:
 [Unknown]: Università di Padova
Specificare la località:
 [Unknown]: Padova
Specificare la provincia:
 [Unknown]: Padova
Specificare il codice a due lettere del paese in cui si
   trova l'unità:
 [Unknown]: IT
Il dato CN=Enrico Righetto, OU=DEI, O=Università di Padova,
L=Padova, ST=Padova, C=IT è corretto?
 [no]: si

Immettere la password della chiave per <tomcat>
    (Invio se corrisponde alla password del keystore):
```

Il comando sopra citato fa riferimento ad un ambiente Windows. In Linux basta sostituire `%JAVA_HOME%:` con `$JAVA_HOME/...` A questo punto si deve modificare il file `TOMCAT_HOME/conf/server.xml` come segue[2]:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector port="8443" minSpareThreads="5" maxSpareThreads="
   75" enableLookups="true" disableUploadTimeout="true"
   acceptCount="100" maxThreads="200" scheme="https" secure
   ="true" SSLEnabled="true" keystoreFile="C:/users/enrico
   /.keystore" keystorePass="password" clientAuth="false"
   sslProtocol="TLS" />
```

Per verificare che il certificato appena creato si integri correttamente con Tomcat, è sufficiente accertarsi che, digitando nel browser l'indirizzo `https://localhost:8443/`, venga visualizzata l'homepage di Tomcat (la stessa di figura 12.2).

## 12.5 Installazione di PostgreSQL

L'installazione di PostgreSQL è molto semplice e non presenta alcuna differenza tra la versione per ambiente Windows e quella per Linux. L'ultima versione del software, al momento della scrittura della presente tesi, è la 8.4.2. L'archivio

[2] In ambiente windows il file `.keystore` viene creato nella cartella `C:/users/nomeutente/`, mentre in Linux il file va cercato in `HOME`

scaricato dal sito internet del produttore presenta un unico file che è sufficiente avviare per far partire la procedura di installazione. Dopo un messaggio di benvenuto, viene chiesto di specificare prima il percorso della directory in cui installare i files dell'applicazione, poi la cartella in cui verranno salvati i database creati. Proseguendo con l'installazione viene, quindi, richiesto l'inserimento della password che si vuole adottare per il *superuser* `postgres` del DBMS. Al passo successivo si richiede, invece, di specificare la porta di comunicazione del DBMS; il consiglio è quello di lasciare il valore predefinito `5432`. Prima di iniziare con la copia dei files su disco viene visualizzata una schermata nella quale scegliere delle opzioni avanzate per utenti esperti; anche qui si consiglia di lasciare i valori predefiniti. Al termine della copia dei files viene avviata un'applicazione che permette di installare alcuni pacchetti affini a PosgreSQL. Per poter utilizzare l'applicazione WQF è necessario installare i drivers JDBC che permettono a Java di Tomcat di comunicare con la base di dati.

### 12.5.1 Caricamento della base di dati

Il database per l'applicazione WQF è disponibile nel Cd-Rom allegato alla tesi come file `WQF.sql`. Esso è stato creato attraverso la procedura per effettuare i backup fornita dall'applicazione `pgAdmin`, che rappresenta l'interfaccia tra il DBMS e l'utente, e contiene solamente i dati minimi utili al funzionamento dell'applicazione.

La prima cosa da fare è creare un nuovo database vuoto di PostgreSQL chiamandolo, ad esempio, `wqf` ed indicandone l'utente proprietario (per quanto concerne questa guida si è assegnato il database all'utente `postgres`). A questo punto si devono caricare i dati nel database a partire dal file di backup. Per fare ciò, dalla versione 8.4, è sufficiente, una volta selezionato il database da popolare, cliccare sul pulsante `PSQL Console` nella barra degli strumenti e digitare, ad esempio, il comando:

```
1 \i C:/WQF.sql
```

Si deve, in pratica, solamente indicare il file da cui caricare i comandi SQL per "costruire" la base di dati. Nelle versioni più vecchie di PostgreSQL non è presente un collegamento diretto all'applicazione `psql` direttamente da `pgAdmin`. Per poter eseguire la procedura di ripristino da backup si deve, allora, avviare manualmente l'applicazione da riga di comando:

```
1 C:\Program Files (x86)\PostgreSQL\8.3\bin>psql -U postgres
    wqf
```

Il comando dice di avviare l'applicazione `psql` connettendosi al database `wqf` come utente `postgres`. Dopo aver inserito la password, se richiesta, è sufficiente digitare il comando descritto precedentemente.

## 12.6 Prima esecuzione dell'applicazione WQF

Per effettuare la prima esecuzione dell'applicazione WQF è necessario utilizzare Tomcat Manager, accessibile attraverso il link presente nel pannello di amministrazione dell'home page di Tomcat (figura 12.2). Dopo aver inserito le credenziali dell'amministratore di Tomcat, scorrere la pagina fino a trovare la voce "Select WAR file to upload" (vedi figura 12.3) dove si deve selezionare il file `.war` relativo all'applicazione; Eclipse permette, infatti, di esportare l'intero codice dell'applicazione in un unico file (con estensione `.war` appunto) caricabile direttamente da Tomcat per effettuare il *deploy* dell'applicazione. Una volta

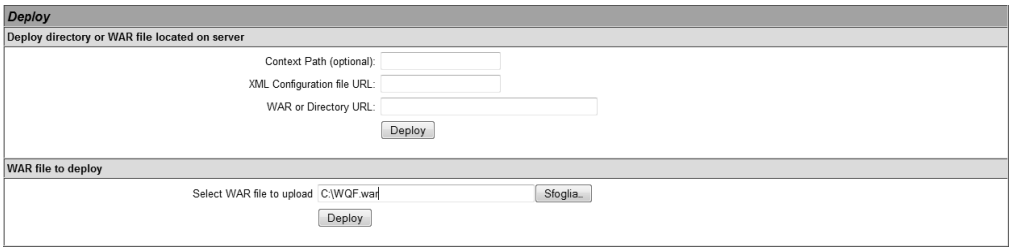


**Figura 12.3.** Deploy di un file `.war` con Tomcat.

effettuata l'operazione di *deploy*, viene creata, in automatico, una cartella `WQF` in `TOMCAT_HOME/webapps` in cui è possibile trovare l'intera struttura dell'applicazione. L'ultima cosa da fare è modificare il file `dataAccessContext.xml` contenuto in `TOMCAT_HOME/webapps/WQF/WEB-INF` inserendo le credenziali per l'accesso al database come riportato nell'esempio seguente:

```
1 <bean id="dataSource"
2         class="org.springframework.jdbc.datasource.
             DriverManagerDataSource">
3         <property name="driverClassName" value="org.
             postgresql.Driver" />
4         <property name="url" value="jdbc:postgresql
             ://127.0.0.1:5432/wqf" />
5         <property name="username" value="postgres" />
6         <property name="password" value="password" />
7     </bean>
```

L'applicazione sarà, quindi, disponibile all'indirizzo `http://localhost:8080/WQF`.

# Capitolo 13

# Manuale del Process Designer

Le funzioni descritte in questo capitolo sono accessibili dal menu `Sviluppo baseline`. Si ricorda che questa sezione è visibile solamente agli utenti che presentano, tra i ruoli di sistema, quello di process designer (`ROLE_DESIGNER`).

## 13.1 Operazioni su baseline

### 13.1.1 Crea baseline

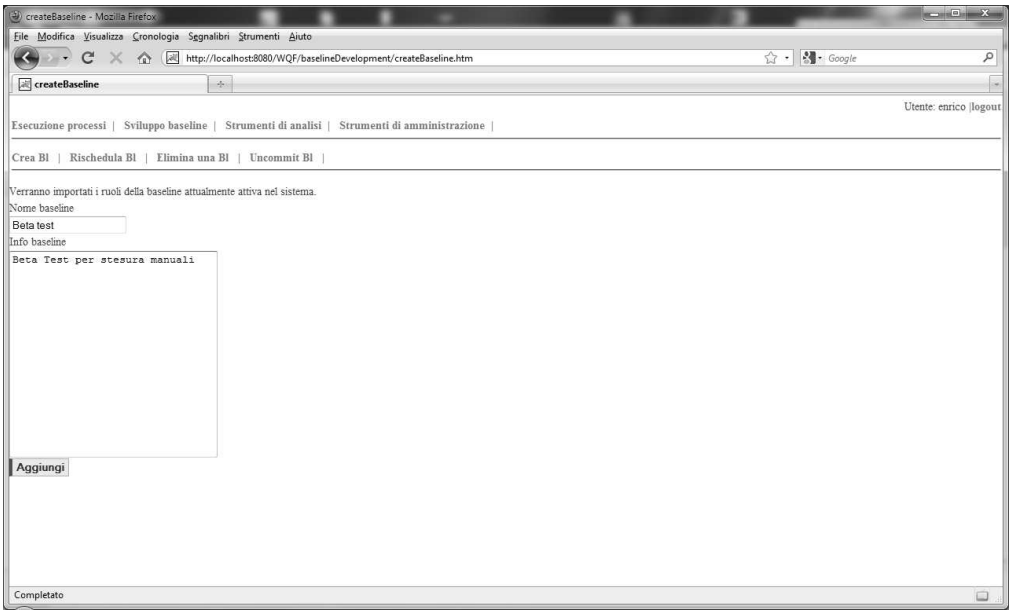


**Figura 13.1.** Creazione di una nuova baseline.

Come illustrato in figura 8.7, quando si esegue per la prima volta l'appli-

cazione WQF si deve, innanzitutto, creare una nuova baseline per portare il sistema allo stato *baseline development*. Una nuova baseline può essere creata accedendo alla pagina `Crea Bl` rappresentata in figura 13.1. Per effettuare l'operazione è sufficiente inserire il nome della baseline e, facoltativamente, una descrizione per, poi, premere il tasto `Aggiungi`; il sistema visualizza un messaggio di conferma nel caso l'operazione vada a buon fine (per l'esempio della figura si potrà leggere "`La baseline ‘‘Beta test’’ è stata creata ed è in stato di sviluppo`"). Nel caso vi sia già una baseline in sviluppo non viene permessa la creazione di una nuova; accedendo alla pagina viene visualizzato un messaggio del tipo "`La baseline Beta test è gia in stato development`".

### 13.1.2 Rischedula baseline

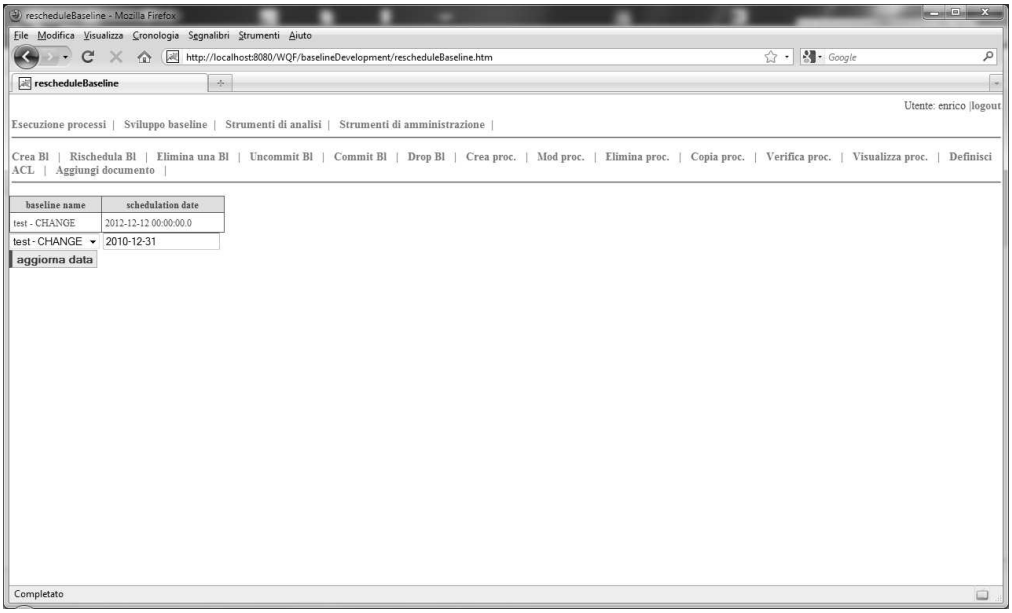


**Figura 13.2.** Rischedulazione di una baseline.

La funzione `Rischedula Bl`, vedi figura 13.2, permette, appunto, di rischedulare una baseline. Nella pagina viene visualizzata la lista delle baseline che si trovano nello stato *committed* ma *non in stato di esecuzione*. Per ognuna di esse viene, inoltre, mostrata la data in cui devono essere attivate. Per rischedulare una baseline è sufficiente sceglierla dal menu a tendina, inserire la nuova data di attivazione e premere il pulsante `Aggiorna data`. Il sistema si occupa di verificare che sia stata inserita una data valida.

### 13.1.3 Elimina baseline

Accedendo alla pagina `Elimina Bl`, è possibile eliminare una delle baseline che si trovano in stato *committed*. La pagina riporta la lista delle baseline eliminabili; per effettuare l'operazione è necessario selezionare la baseline desiderata dal menu a tendina e premere il pulsante `Delete baseline`. Non è stata volutamente riportata l'immagine della pagina, in quanto essa è strutturata in maniera del tutto simile a quella per la rischedulazione di una baseline.

### 13.1.4 Uncommit baseline

La pagina `Uncommit Bl` permette di riportare allo stato *development* una baseline *committed* a patto che non vi sia già una baseline in sviluppo. In questo caso, infatti, viene visualizzato il messaggio `Esiste già una baseline in sviluppo`. Al contrario, se è possibile effettuare l'operazione, viene visualizzato l'elenco delle baseline *committed* e viene chiesto di selezionare la baseline da riportare in sviluppo selezionandola da un menu a tendina per poi premere il pulsante `Uncommit baseline`. Anche questa pagina rispetta il layout di quelle descritte in precedenza e non è stata, quindi, ritenuto necessario inserire la relativa immagine.

### 13.1.5 Commit baseline

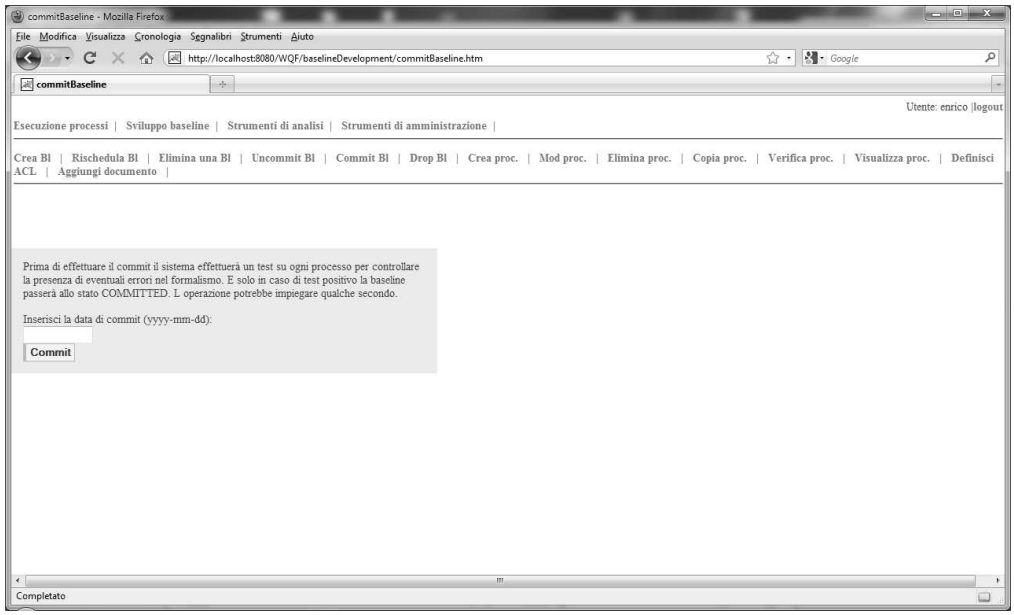


**Figura 13.3.** Pagina per effettuare il commit di una baseline.

Attraverso la funzione `Commit Bl` è possibile, appunto, portare la baseline in sviluppo allo stato *committed*. Nella pagina, vedi figura 13.3, viene ricordato all'utente che il sistema, prima di effettuare il commit, effettuerà un test su ogni processo per controllare la presenza di eventuali errori nel formalismo (vedi la funzione *verifica processo*). Solo in caso di test positivo la baseline passerà allo stato *committed*. L'utente deve solamente indicare la data in cui dovrà essere attivata la baseline e premere il pulsante `Commit`. Eventuali errori nel disegno dei processi vengono visualizzati nella pagina stessa al termine dell'analisi.

### 13.1.6 Drop baseline

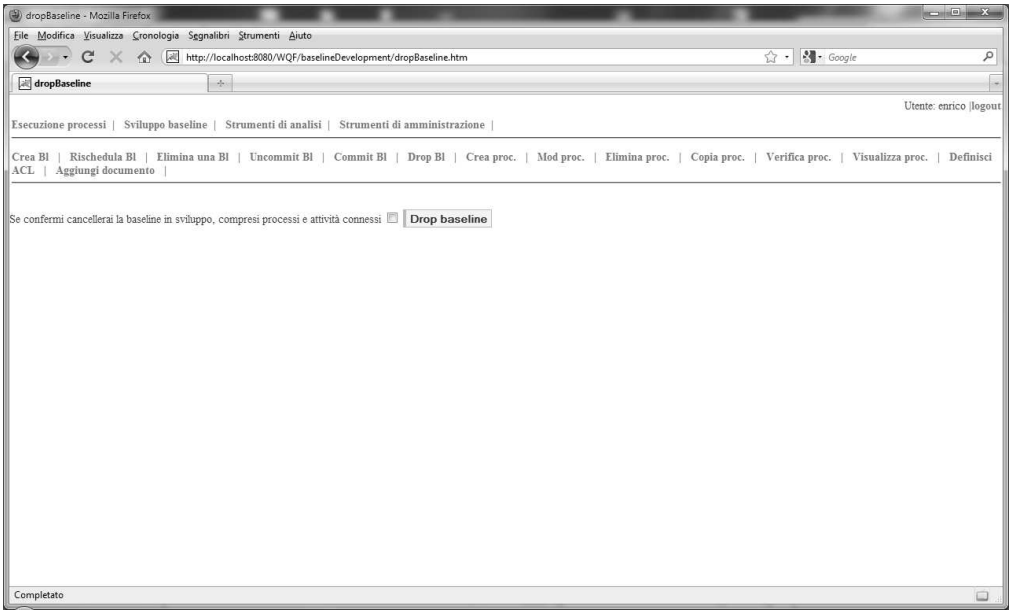


**Figura 13.4.** Drop della baseline in stato di sviluppo.

La pagina `Drop Bl`, figura 13.4, permette di eliminare tutti i dati relativi alla baseline in sviluppo. L'utente viene informato che, con l'operazione, verranno persi tutti i dati (processi e attività connessi) quindi, per continuare con l'operazione gli viene richiesto di confermare il tutto spuntando la checkbox alla fine del messaggio e premere il pulsante `Drop baseline` (altrimenti disabilitato).

## 13.2 Operazioni su processi

### 13.2.1 Crea processo

Per creare un nuovo processo, il disegnatore deve accedere alla pagina `Crea proc.` visualizzata in figura 13.5. La procedura prevede l'inserimento di un

nome del processo e di una descrizione (facoltativa). Il processo di inserimento termina con la pressione del pulsante `Aggiungi`. Se l'operazione di inserimento va a buon fine viene visualizzato un messaggio di conferma come, ad esempio, "`Processo ‘‘Product Planning’’ creato con successo.`". Il sistema si occupa di verificare l'unicità del nome inserito. Nel caso si tenti di creare un nuovo processo utilizzando il nome di un processo già esistente, viene, invece, mostrato un messaggio del tipo "`Esiste già un processo ‘‘Product Planning’’ in questa baseline.`"

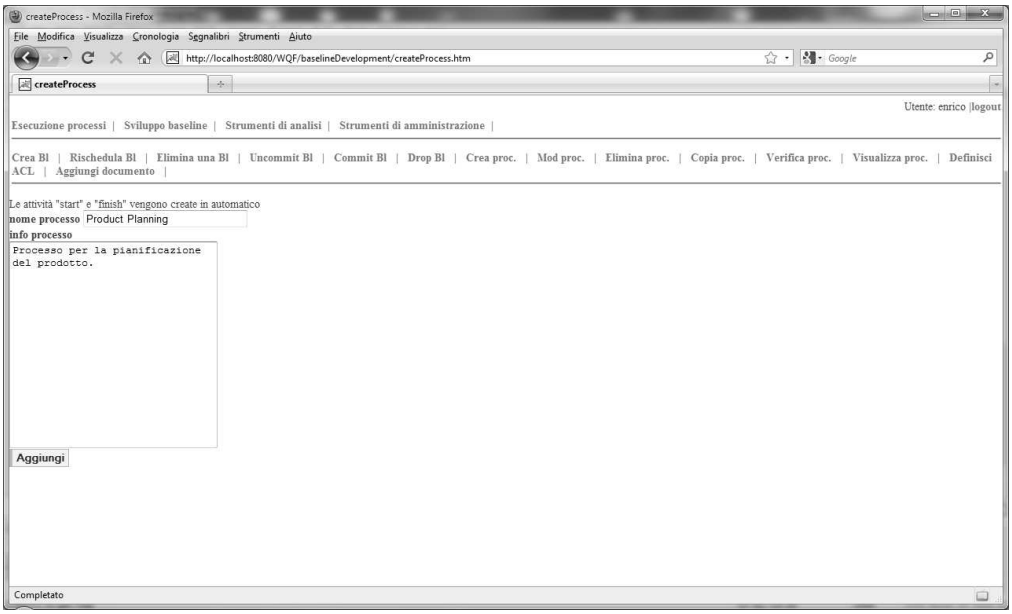


**Figura 13.5.** Creazione di un nuovo processo.

### 13.2.2 Modifica processo

Accedendo alla pagina `Mod proc.`, vedi figura 13.6, è possibile modificare le informazioni (nome e descrizione) di un processo precedentemente creato. L'utente deve selezionare il processo dalla lista sulla sinistra, ricompilare il form con le nuove informazioni (il sistema carica di default quelle precedentemente inserite) e premere il pulsante `Modifica processo`. Accedendo a questa pagina viene, inoltre, espanso il menu delle funzioni disponibili visualizzando tutte le operazioni possibili su un processo. Si ricorda, infatti, che, al momento della sua creazione, un nuovo processo è, di fatto, una scatola vuota con solamente l'attività *start* e quella *finish*. Nei paragrafi seguenti verranno illustrate le procedure da seguire per utilizzare ognuna delle funzioni di editing dei processi.

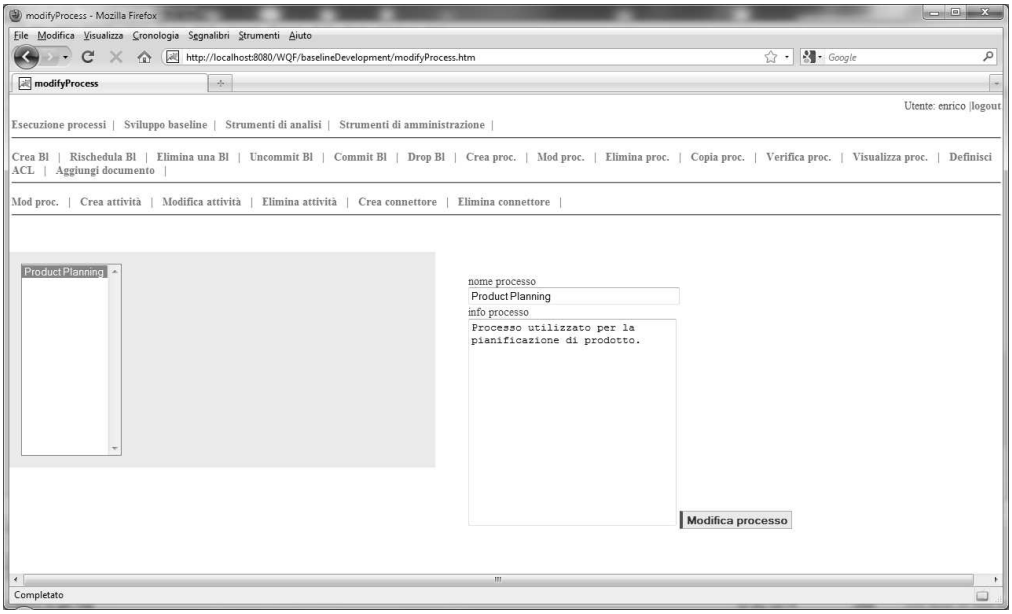


**Figura 13.6.** Modifica di un processo.

## Crea attività

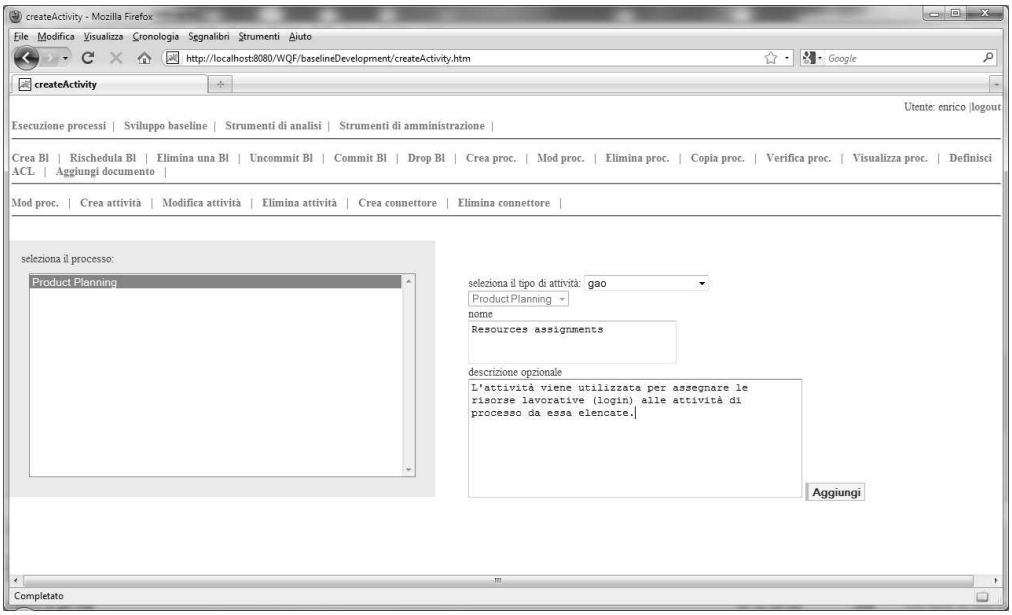


**Figura 13.7.** Creazione di una nuova attività.

Mediante la pagina `Crea attività`, di cui è riportato un esempio in figura 13.7, è, appunto, possibile creare una nuova attività per un certo processo precedentemente creato. Per prima cosa, l'utente deve selezionare il processo a cui l'attività fa riferimento. In seguito si devono indicare rispettivamente il tipo, il nome e la descrizione (opzionale) dell'attività. Nel caso si selezioni il tipo *Subprocess* si deve, inoltre, indicare anche il sottoprocesso da attivare selezionandolo tra quelli già presenti nella base di dati. Per terminare l'operazione premere il pulsante `Aggiungi`. Anche per le attività valgono le regole di unicità del nome valide per i processi; il sistema comunica l'avvenuto inserimento o la presenza di errori di compilazione del form con modalità simili a quelle precedentemente descritte.

### Modifica attività

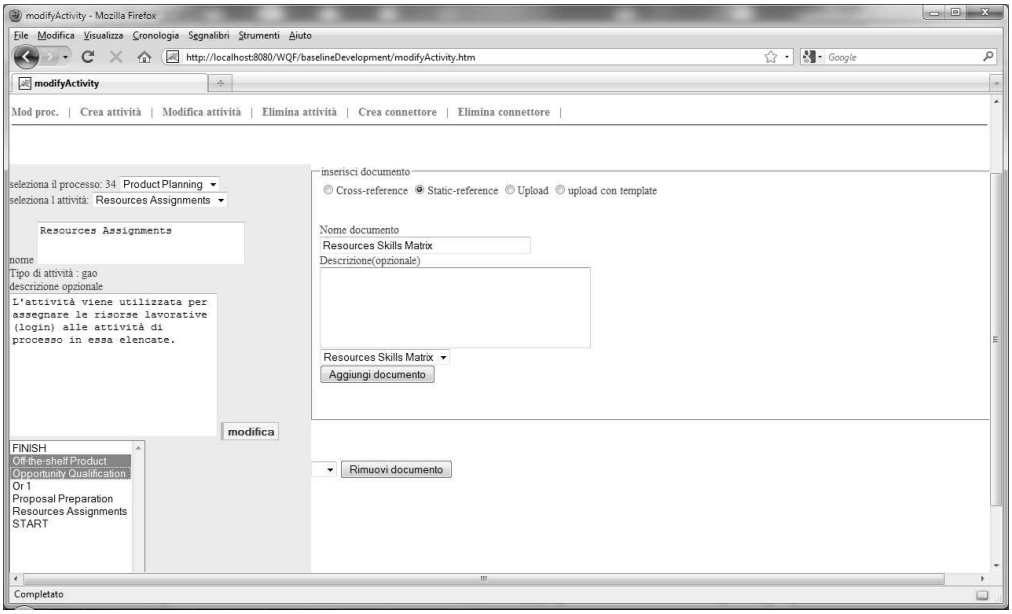


**Figura 13.8.** Modifica di un'attività.

Come avviene con i processi, anche le attività vengono create come delle box vuote. Per modificarle, si deve accedere alla pagina `Modifica attività`. L'esempio in figura 13.8 mostra la modifica di un'attività di tipo GAO; è stato volutamente scelto questo tipo di attività perchè la pagina richiede l'inserimento di un numero maggiore di informazioni.

Dopo aver selezionato il processo di riferimento e l'attività da modificare, nella parte sinistra della pagina viene visualizzato un form nel quale si possono modificare i valori di nome e descrizione precedentemente inseriti. Per le attività di tipo GAO come quella dell'esempio, viene, inoltre, visualizzata la lista

delle attività per il processo scelto dalla quale si devono selezionare quelle che saranno da essa gestite (si ricorda che la scelta multipla è effettuabile attraverso il comando `ctrl+click`). Premendo il tasto `Modifica` le informazioni vengono aggiornate.

La parte destra della pagina, invece, viene utilizzata per inserire le informazioni riguardanti i documenti di input/output. In base alla tipologia di documento da collegare all'attività, viene richiesto l'inserimento di informazioni differenti:

- *Cross-reference*. L'utente deve indicare l'attività che produce il documento, selezionarlo tra le opzioni possibili ed indicare un nome ed una descrizione (facoltativa) che verranno, poi, visualizzati nella pagina per l'esecuzione dell'attività in modifica.

- *Static-reference*. L'utente deve indicare, come al punto precedente, nome e descrizione del documento e selezionare il riferimento tra i documenti disponibili nel menu a tendina. Il documento da collegare deve essere stato precedentemente inserito usando la funzione per l'aggiunta nel sistema di documenti/template in seguito illustrata.

- *Upload*. Viene richiesto di inserire il nome ed, opzionalmente, una descrizione del documento.

- *Upload con template*. Si richiede di inserire i dati relativi al documento (nome e descrizione facoltativa), nonchè il riferimento al template (precaricato) come nel caso delle *Static-reference*.

Per concludere la procedura di inserimento di un documento si deve premere il pulsante `Aggiungi documento`. Nella parte inferiore della pagina viene riportata la lista dei documenti già inseriti per l'attività in modifica.

### Elimina attività

Per eliminare un'attività dal disegno di un processo, si deve accedere alla pagina `Elimina attività`. Selezionando il processo dall'elenco a sinistra, viene automaticamente aggiornata la lista delle attività precedentemente create per esso. È, quindi, possibile eliminare un'attività selezionandola e agendo sul pulsante `Delete activity`. Si ricorda che le attività "START" e "FINISH" non possono essere eliminate in quanto create in automatico dal sistema, perciò non compariranno nella lista.

### Crea connettore

Le connessioni tra le diverse attività di un processo possono essere definite accedendo alla pagina `Crea connettore`. Una volta selezionato il processo da

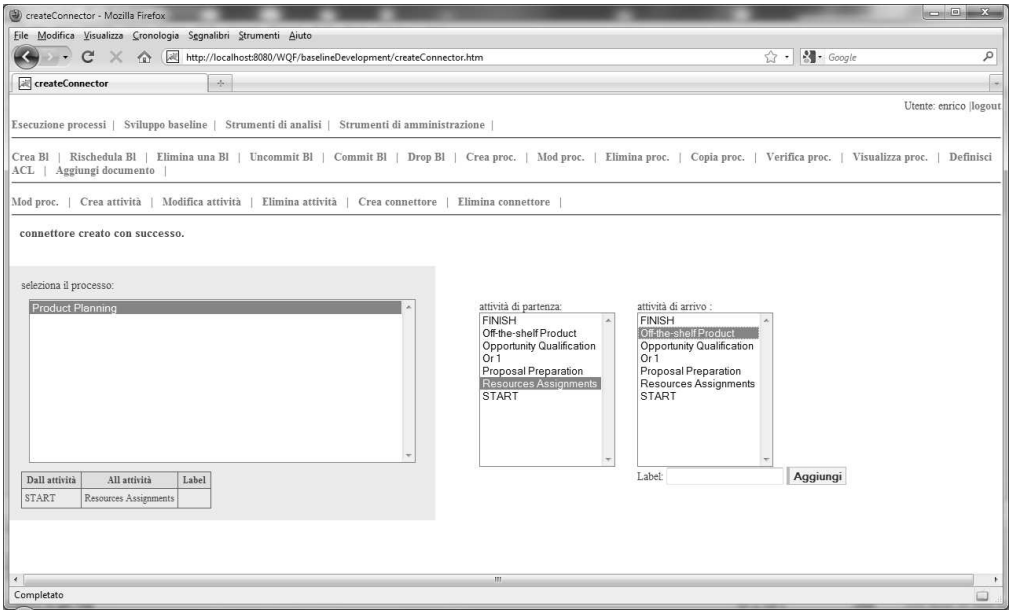


**Figura 13.9.** Pagina per la creazione dei connettori tra le attività di un processo.

modificare (elenco a sinistra), si devono indicare l'attività da cui parte il connettore e quella di arrivo dalle due liste popolate automaticamente sulla destra (vedi figura 13.9). L'operazione si conclude con la pressione del pulsante `Crea connettore`. A creazione avvenuta viene visualizzato un messaggio di conferma del tipo "`Connettore creato con successo`". Mano a mano che i connettori vengono inseriti, viene aggiornata la tabella dei connettori già presenti visibile a fondo pagina. Si ricorda che i connettori uscenti da attività di scelta devono necessariamente essere etichettati. La label può essere inserita nell'apposita casella di testo. nel caso si tenti di creare un connettore uscente da una scelta privo di etichetta, il sistema lo segnalerà con il messaggio di errore "`Non hai inserito la label per un attività di tipo ‘‘scelta’’`".

### Elimina connettore

La procedura per eliminare uno o più connettori è disponibile alla pagina `Elimina connettore` rappresentata in figura 13.10. Qui si deve, per prima cosa selezionare il processo da modificare; in questo modo il sistema genera, in automatico, la lista di tutti i connettori precedentemente creati. A questo punto è sufficiente selezionare il/i connettore/i da eliminare (selezionando le checkbox) e premere il pulsante `Delete`.

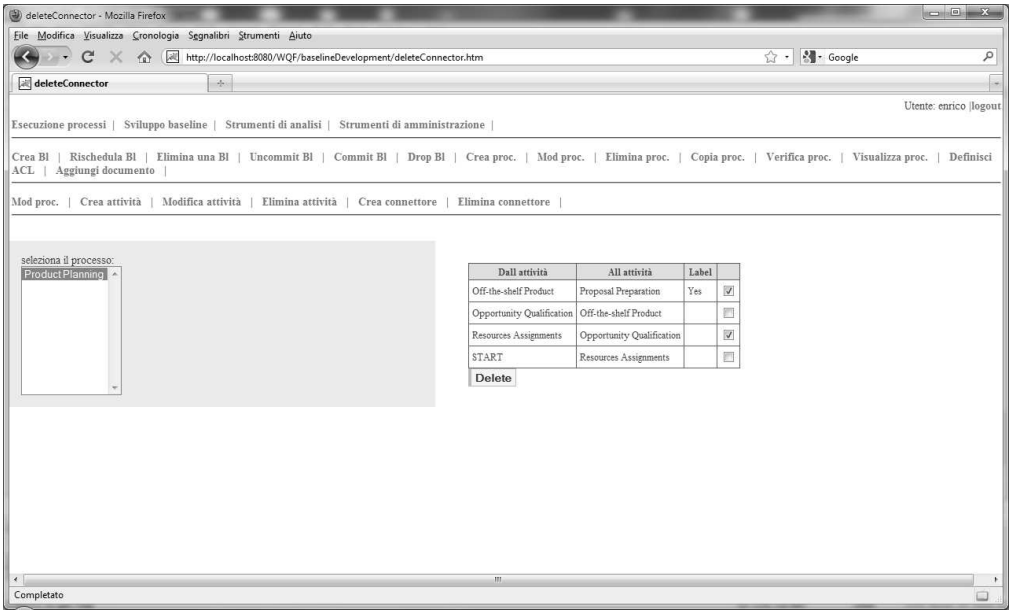


**Figura 13.10.** Eliminazione di uno o più connettori.

### 13.2.3 Elimina processo

Mediante la pagina `Elimina processo` è, appunto, possibile eliminare processi creati nella baseline in sviluppo. La procedura consiste nel selezionare il processo da eliminare dalla lista visualizzata e premere il pulsante `Delete process`. Il buon esito dell'operazione viene segnalato con il messaggio "`Processo eliminato`". Si ricorda che l'eliminazione di un processo comporta la conseguente cancellazione di attività e connettori ad esso affini.

### 13.2.4 Copia processo

Se si desidera recuperare un processo di una vecchia baseline per inserirlo nella baseline attualmente in sviluppo, è sufficiente accedere alla pagina `Copia proc.` (vedi figura 13.11). In essa vengono visualizzati tutti i processi presenti nella base di dati suddivisi per baseline. L'utente deve solamente selezionare le checkbox relative ai processi che intende recuperare e premere il pulsante `Copia processi`. Il sistema copia attività e connettori dei singoli processi. Non vengono, invece, recuperati eventuali sottoprocessi, documenti *cross-reference* e le informazioni di ACL.

### 13.2.5 Verifica processo

Prima di concludere il disegno di un processo, è consigliabile effettuare una verifica di compatibilità con il formalismo per la descrizione dei processi. Questa

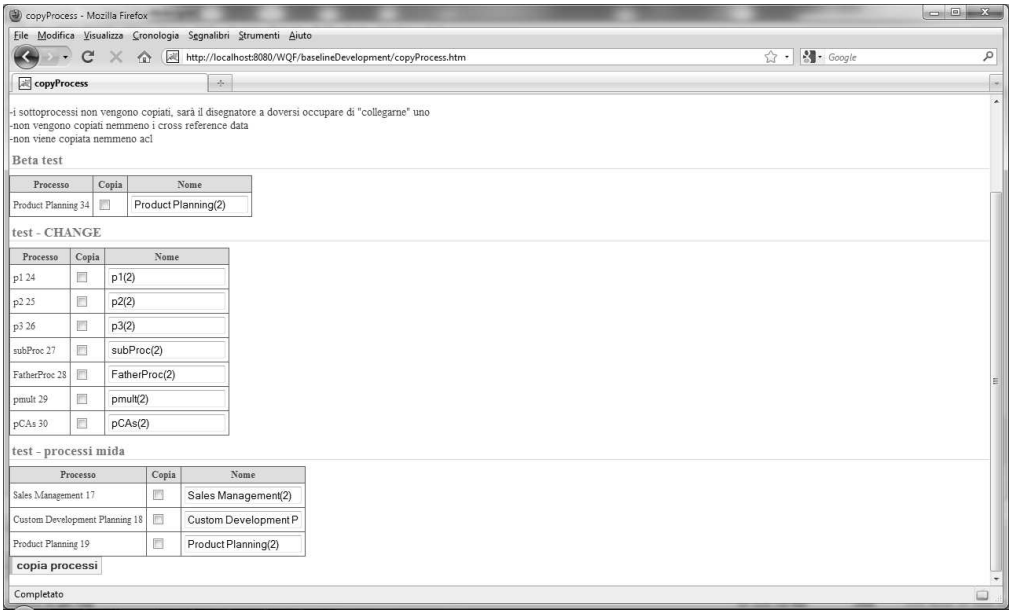


**Figura 13.11.** Copia di un processo da un'altra baseline.

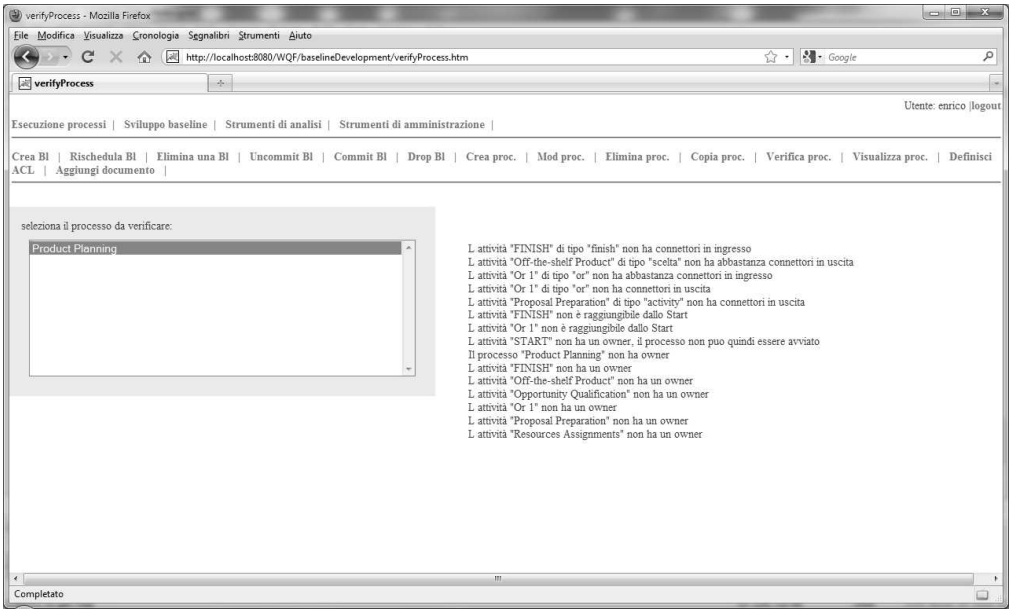


**Figura 13.12.** Verifica della presenza di errori di formalismo in un processo.

funzione è disponibile alla pagina `Verifica proc.` (vedi figura 13.12) e coincide perfettamente con la funzione chiamata dal sistema al momento del commit

di una baseline. In seguito alla selezione da parte dell'utente del processo da verificare, vengono visualizzate tutte le eventuali incongruenze con il formalismo. I controlli effettuati sono:

- verifica che l'attività Finish sia raggiungibile;
- verifica che ogni attività Scelta abbia almeno due connettori in uscita;
- verifica che ogni attività Or abbia almeno due connettori in ingresso;
- verifica che ogni attività sia raggiungibile dallo Start;
- verifica che sia stato definito un owner per il processo;
- verifica che sia stato definito un owner per ogni attività.

### 13.2.6 Visualizza processo

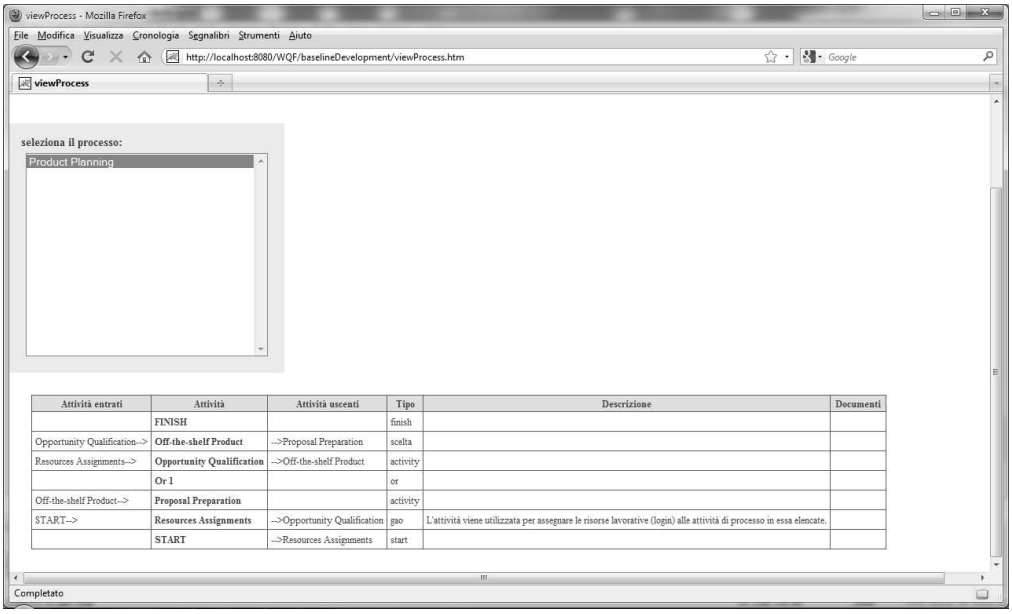


**Figura 13.13.** Visualizzazione di un processo.

Nel disegnare i processi è, spesso, utile visualizzare il lavoro svolto. La pagina `Visualizza proc.`, vedi figura 13.13, permette di verificare quali informazioni sono state inserite e quali no. Selezionando un processo tra quelli disegnati, viene generata una tabella che mostra, per ogni attività, chi la precede, chi la segue, il tipo, l'eventuale descrizione ed i documenti ad essa riferiti.

## 13.3 Definizione Access Control List

Attraverso la pagina `Definisci ACL`, vedi figura 13.14, il Process Designer definisce i permessi su processi e attività per ognuno dei ruoli aziendali.

La prima tabella che viene costruita rappresenta l'Access Control List per i processi; in essa vi è una colonna per ogni ruolo. Qui l'utente seleziona il processo per cui saranno definiti i permessi (anche per le sue attività, come verrà spiegato a breve). La compilazione della tabella dei permessi sul processo richiede obbligatoriamente soltanto l'indicazione di almeno un ruolo come owner del processo.

La seconda tabella rappresenta, invece, l'ACL per le singole attività del processo selezionato. Vi è una riga per ogni attività, mentre le colonne, come nella tabella di processo, corrispondono ai ruoli aziendali. Se l'ACL di un'attività rispecchia in toto quella del processo, è possibile selezionare la checkbox nella colonna `Eredita` e premere il pulsante `Aggiorna activity ACL`: tutte le righe con la checkbox selezionata rispecchieranno la struttura dei permessi del processo e l'ACL verrà salvata. Il pulsante `Reload` può essere utilizzato, invece, per annullare modifiche non ancora salvate e ricaricare l'ACL. Si ricorda che anche le attività devono avere almeno un ruolo con permesso owner. Il pulsante `Aggiorna activity ACL` deve essere premuto al termine della compilazione per rendere definitivi i cambiamenti. Essendo la compilazione della tabella un lavoro lungo e delicato, si consiglia di effettuare salvataggi periodici.

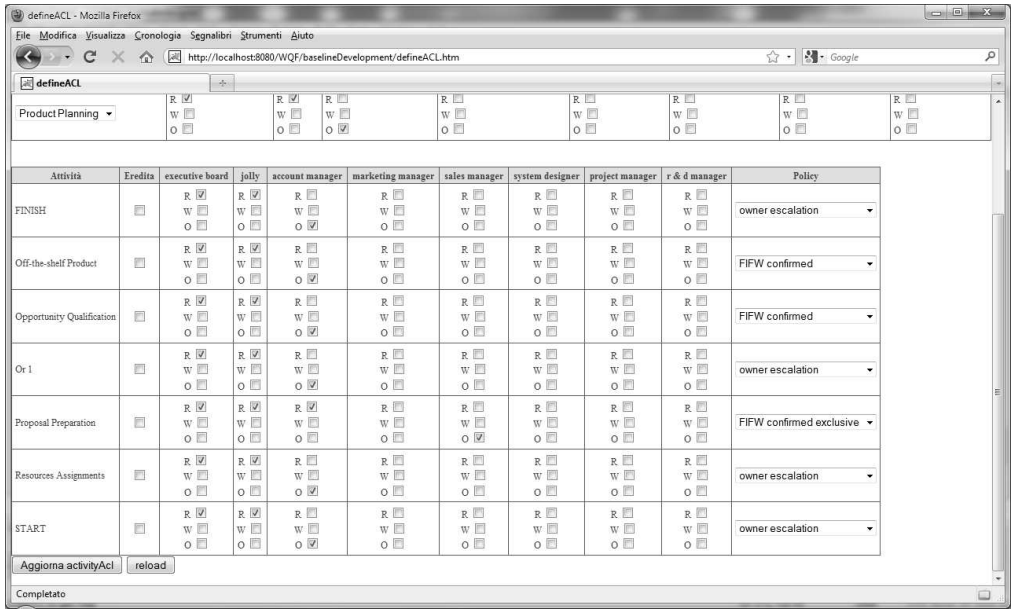


**Figura 13.14.** Access Control List: gestione dei permessi su processi e attività.

## 13.4 Aggiunta documenti e template

Il sistema mette a disposizione una procedura per l'inserimento dei documenti statici e dei template da utilizzare nell'esecuzione delle diverse attività. Questa è accessibile alla pagina `Aggiungi documento` rappresentata in figura 13.15. Dopo aver indicato se si intende inserire un documento o un template (vengono salvati in differenti tabelle della base di dati), l'utente deve indicarne il nome, l'URL da dove recuperarlo, la data di creazione ed, opzionalmente, il numero di revisione e una nota descrittiva. La procedura si conclude con la pressione del pulsante `Aggiungi`. Un inserimento avvenuto con successo viene segnalato con il messaggio "`Documento inserito`".

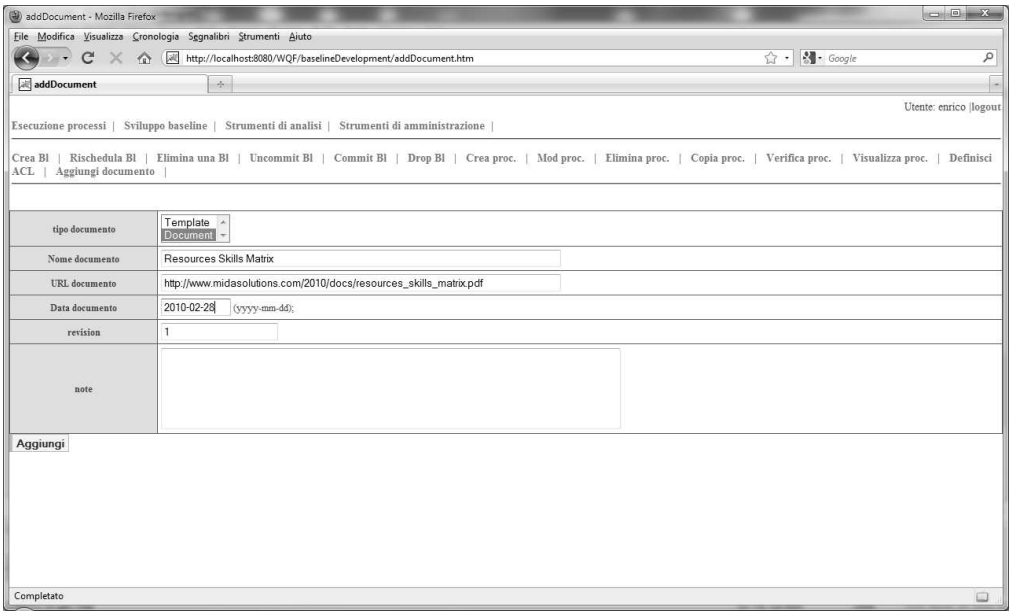


**Figura 13.15.** Aggiunta di un documento o template.

# Capitolo 14

# Manuale utente

## 14.1 Esecuzione processi

### 14.1.1 Esecuzione di un'attività

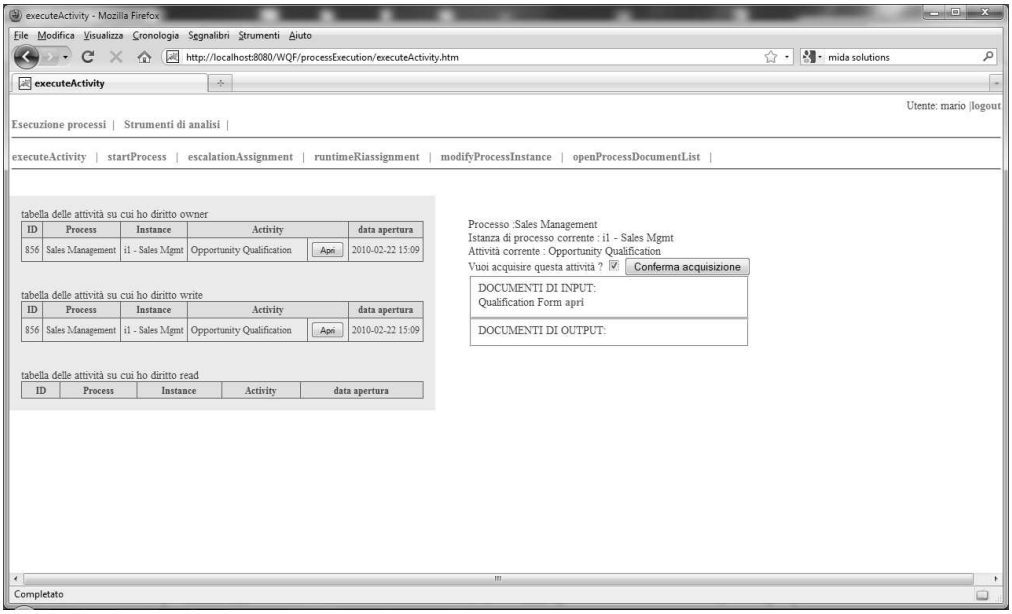


**Figura 14.1.** Accettazione di un'attività.

Cliccando su `Execute activity` ogni utente vede, sulla sinistra, l'elenco delle attività da svolgere, suddivise in base al permesso ottenuto per ognuna di esse (owner, write e read). Cliccando sul pulsante `Apri` vicino al nome di un'attività, sulla parte destra della pagina compariranno i dettagli relativi all'attività selezionata. Per poter eseguire le istruzioni è, però, necessario confermare l'accettazione dell'attività, vedi figura 14.2, in caso questa sia stata definita con

politica FIFW. Una volta acquisita, l'utente può svolgere il lavoro sull'attività e, in base ai permessi accordati, può leggere e/o caricare documenti. Una volta concluso il lavoro sull'attività (devono essere stati caricati tutti i documenti richiesti), l'utente con permesso owner potrà chiuderla semplicemente agendo sul pulsante `termina attività`, il quale viene visualizzato solamente non appena è realmente possibile effettuare la chiusura (vedi figura 14.2).

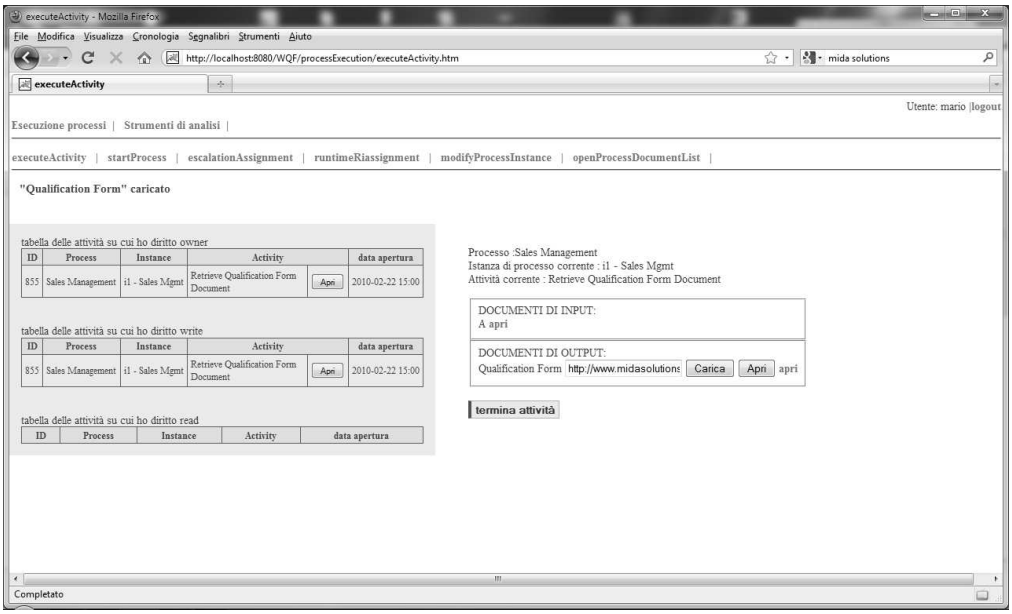


**Figura 14.2.** Upload documenti e chiusura attività.

### 14.1.2 Apertura di una nuova istanza di processo

Nella pagina `Start Process`, un utente può avviare nuove istanza dei processi per i quali dispone del permesso owner sulla relativa attività Start. La struttura della pagina è visualizzata in figura 14.3. Sulla parte sinistra viene visualizzato l'elenco dei processi avviabili; una volta selezionato il processo per cui si vuole aprire una nuova istanza dal menu a tendina in alto a destra, viene richiesto di inserire il nome della nuova istanza. L'inserimento della descrizione è facoltativo. Per avviare il processo si deve agire sul pulsante `Start Process`. Non possono essere presenti più istanze di processo con il medesimo nome, per questo, al momento dell'inserimento, viene effettuato un controllo di unicità del nome. Nel caso in cui il nome digitato sia già stato usato in precedenza, l'operazione di avvio del processo non va a buon fine ed il tutto viene segnalato con un avviso sulla pagina. Per evitare di ripetere l'errore, l'applicazione mette a disposizione una funzione di ricerca, avviabile tramite il pulsante `cerca`, che cerca nella base di dati tutti i nomi delle istanze del processo selezionato che iniziano con il testo

inserito. Se, per esempio, è stato digitato il testo “Ist”, verranno visualizzati risultati del tipo “Istanza 1”, “ist-2”, ecc.

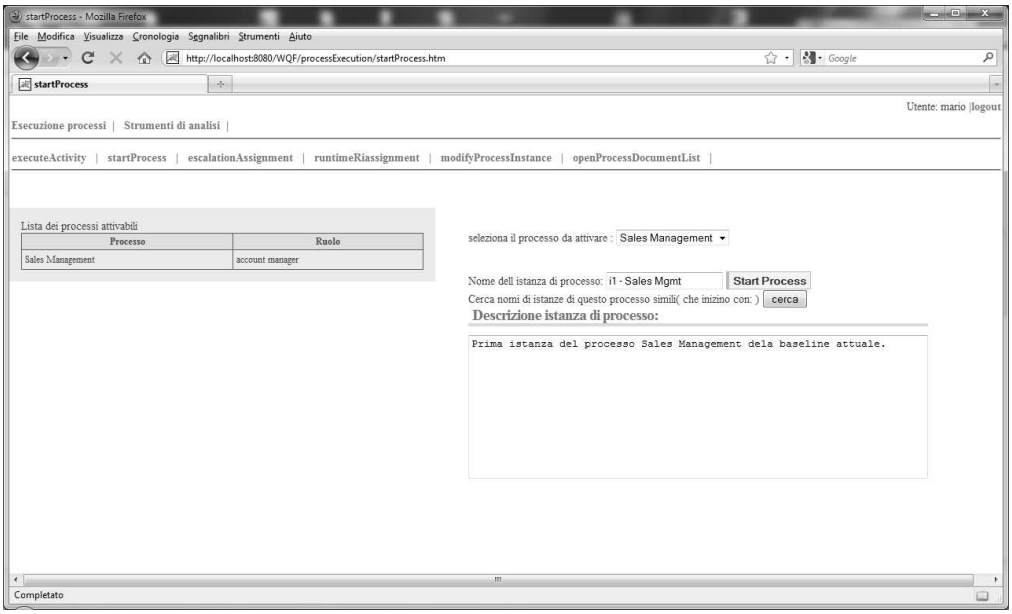


**Figura 14.3.** Apertura di una nuova istanza di processo.

### 14.1.3 Escalation per assegnementi

Quando il sistema apre un attività con politica di assegnazione Owner escalation, esso produce una segnalazione verso la login prestabilita (la persona indicata come assigner per il ruolo con permesso owner sull'attività in esame) richiedendo l'assegnazione delle risorse lavorative. Gli utenti trovano le richieste di assegnamento nella pagina `Escalation assignement` come rappresentato in figura 14.4. Selezionando dalla lista a sinistra l'attività per la quale si desidera effettuare l'assegnamento, compare sulla destra l'elenco dei ruoli che presentano un qualche permesso sull'attività e, per ciascuno di essi, l'elenco delle relative login. L'assegnatore deve selezionare le login a cui assegnare l'attività (la selezione multipla va effettuata con il comando `ctrl+click`. È obbligatorio selezionare almeno una login per il ruolo con permesso owner, altrimenti l'attività resterebbe bloccata.

### 14.1.4 Riassegnamenti runtime

L'assigner di un ruolo può, in ogni momento, modificare gli assegnamenti fatti in precedenza accedendo alla pagina `Runtime riassignment` visualizzata in figura 14.5. La struttura della pagina è del tutto simile a quella degli assenamenti

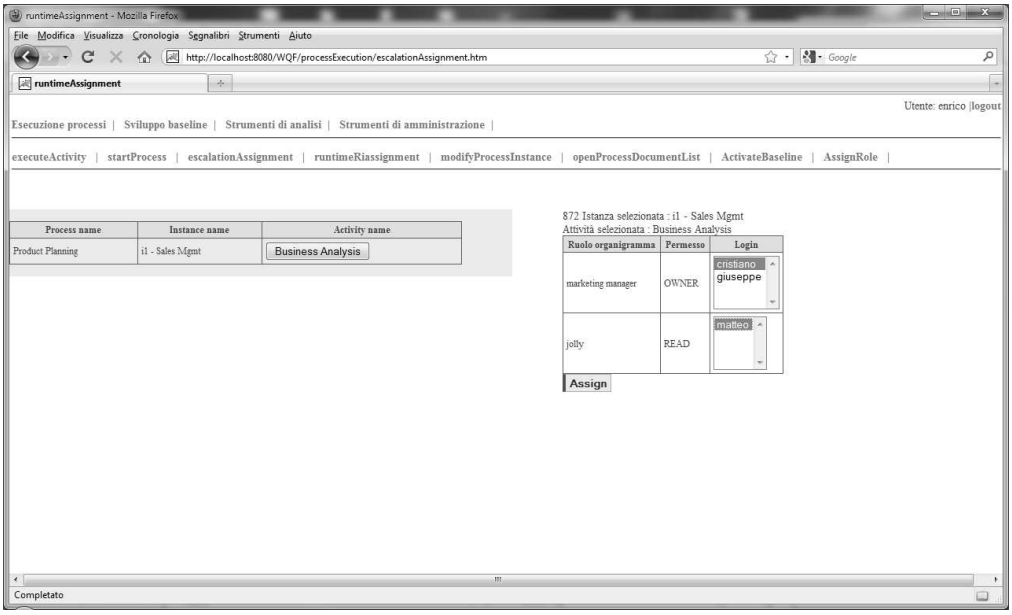


**Figura 14.4.** Assegnamento risorse lavorative in seguito ad un'escalation.

per escalation descritta nella sezione precedente. Selezionando un'attività dalla lista viene visualizzata la tabella per l'assegnamento delle risorse. Viene data la possibilità di modificare i permessi anche per le attività aventi come policy una delle due FIFW (esclusiva e non), a patto che esse non siano ancora state acquisite da un utente. Pre-assegnare una di queste attività toglie la possibilità ad altri utenti di acquisirla. Questo è stato fatto in quanto si presuppone che un'assegnazione da parte dell'assigner abbia una priorità superiore rispetto alle scelte degli utenti.

### 14.1.5 Modifica di un'istanza di processo

Il sistema dà la possibilità agli utenti di modificare le istanze di processo da loro aperte per, ad esempio, rinominarle o cambiarne la descrizione. Questa operazione è disponibile aprendo la pagina `Modify process instance` rappresentata in figura 14.6. La pagina è praticamente identica a quella per la creazione di una nuova istanza. selezionando l'attività da modificare (anzichè il processo da aprire) vengono visualizzati i dati precedentemente inseriti con la possibilità di modificarli. Il controllo sull'unicità del nome avviene come con l'apertura di una nuova istanza.

L'utente owner del processo ha, inoltre, la possibilità di far abortire l'istanza di processo selezionata premendo il pulsante `Abort instance`. In questo caso, quindi, tutte le attività aperte relative all'istanza selezionata verranno, a loro volta, abortite.

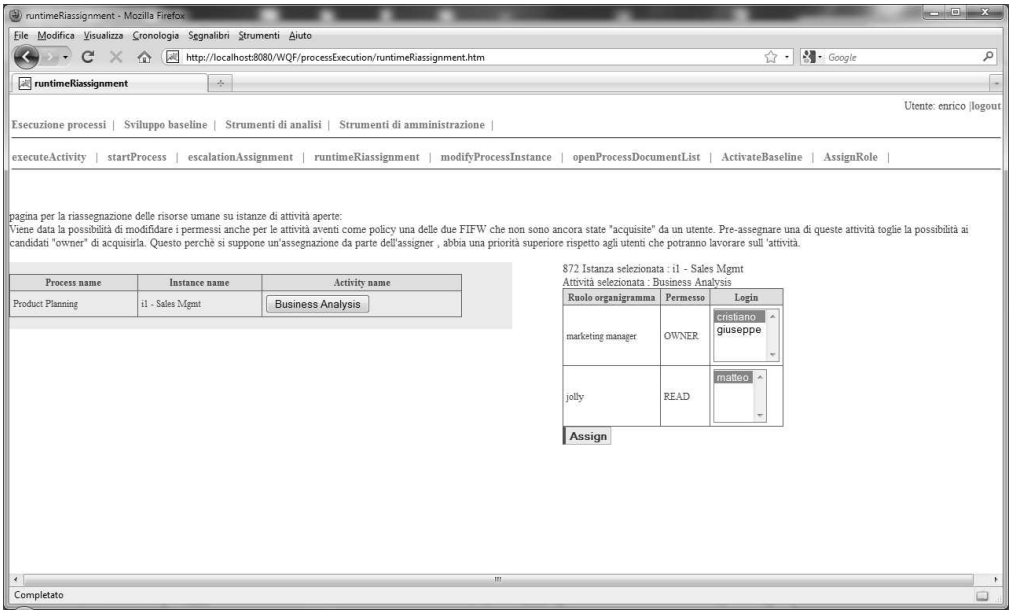


**Figura 14.5.** Rissegnamento risorse lavorative in runtime.

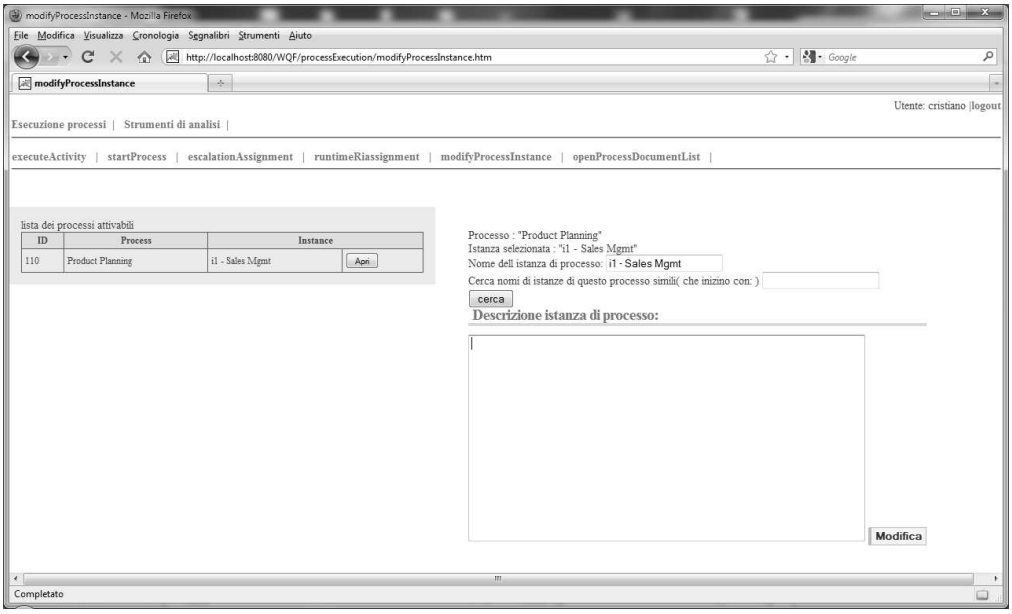


**Figura 14.6.** Modifica di un'istanza di processo.

### 14.1.6 Riepilogo documenti

La pagina `Open process document list` (vedi figura 14.7) permette agli utenti di visualizzare l'elenco dei documenti relativi ad un'istanza di processo attiva. La funzione può risultare molto utile nel caso in cui si debbano prendere alcune decisioni sulla base di risultati prodotti da altre attività e che non sono direttamente consultabili svolgendo l'attività corrente. Nella pagina viene presentata la lista dele istanze di processo attive per le quali l'utente gode di qualche permesso. Selezionando una delle istanze, viene generata la lista dei documenti che afferiscono ad essa. Si tenga presente che se un documento si riferisce ad un'attività per la quale l'utente che sta effettuando la richiesta non presenta alcun permesso esso non sarà presente nella lista.

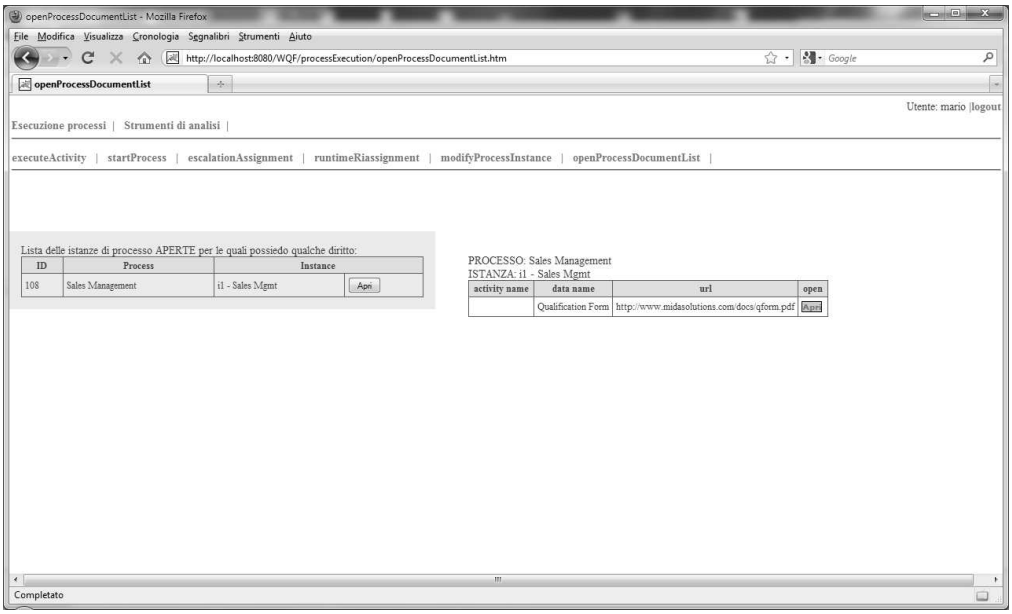


**Figura 14.7.** Visualizzazione dei documenti relativi ad una particolare istanza (attiva) di processo.

## 14.2 Strumenti di analisi

### 14.2.1 Creazione di filtri di ricerca da parte dell'amministratore

Gli utenti con ruolo di sistema ROLE_ADMIN hanno la possibilità di creare dei filtri di ricerca per effettuare l'analisi dei dati per qualsiasi altro utente come visualizzato in figura 14.8. La pagina è accessibile al percorso `Analisi dati > Creazione filtri`. Dopo aver selezionato i campi con cui effettuare l'interrogazione sui dati (show mask)ed i campi su cui visualizzare i risultati (show result)

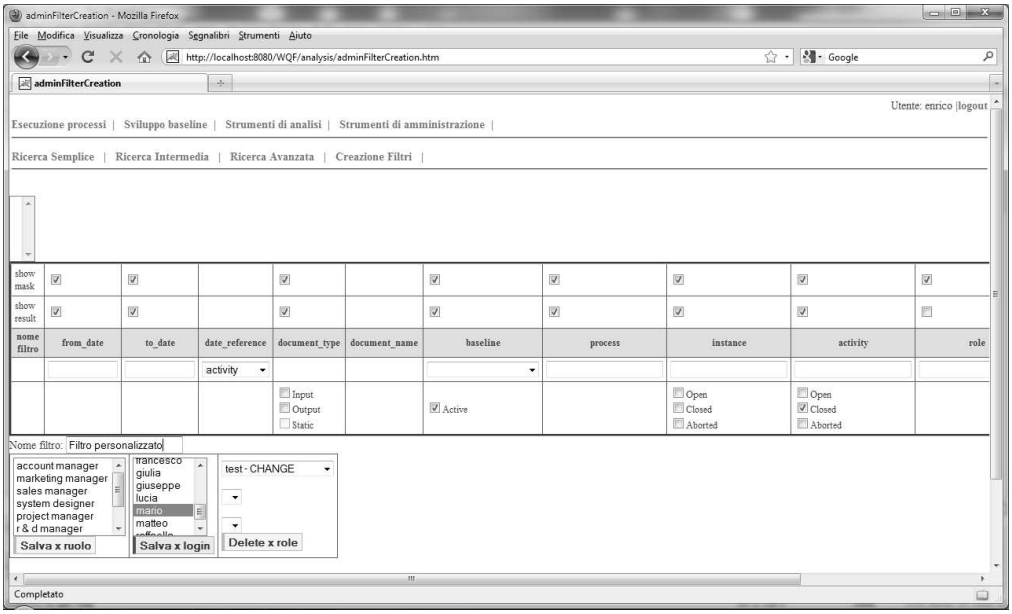


**Figura 14.8.** Creazione di filtri di ricerca da parte dell'amministratore.

ed aver, eventualmente, inserito i valori su cui effettuare la ricerca, è possibile salvare il filtro specificando chi lo visualizzarà e lo potrà, di conseguenza, usare. È possibile salvare un filtro per uno o più ruoli o per le singole login premendo, rispettivamente, sul pulsante `Salva per ruolo` o `Salva per login`. Al momento del salvataggio del filtro il sistema si occupa di memorizzare la login del creatore che sarà l'unica persona che lo potrà eliminare.

### 14.2.2 Ricerca semplice

Nella pagina `Ricerca semplice`, l'utente visualizza la lista dei filtri di ricerca da lui/per lui creati. Qui si deve selezionare un filtro dalla lista premere il pulsante `Cerca`. Vengono, quindi, immediatamente visualizzati i risultati della ricerca nello spazio inferiore della pagina. Nell'intestazione di ogni colonna è presente un bottone. Premendolo, i risultati vengono ordinati con ordine crescente rispetto al campo selezionato. La figura 14.9 mostra i risultati della ricerca effettuata dall'utente "mario" utilizzando il filtro di ricerca "Filtro personalizzato" creato per cercare tutte le attività già chiuse su cui l'utente possedeva un qualche permesso.

### 14.2.3 Ricerca intermedia

La pagina `Ricerca intermedia` consente all'utente di caricare filtri di ricerca esistenti e di modificare i valori dei campi di ricerca. Non è possibile aggiun-

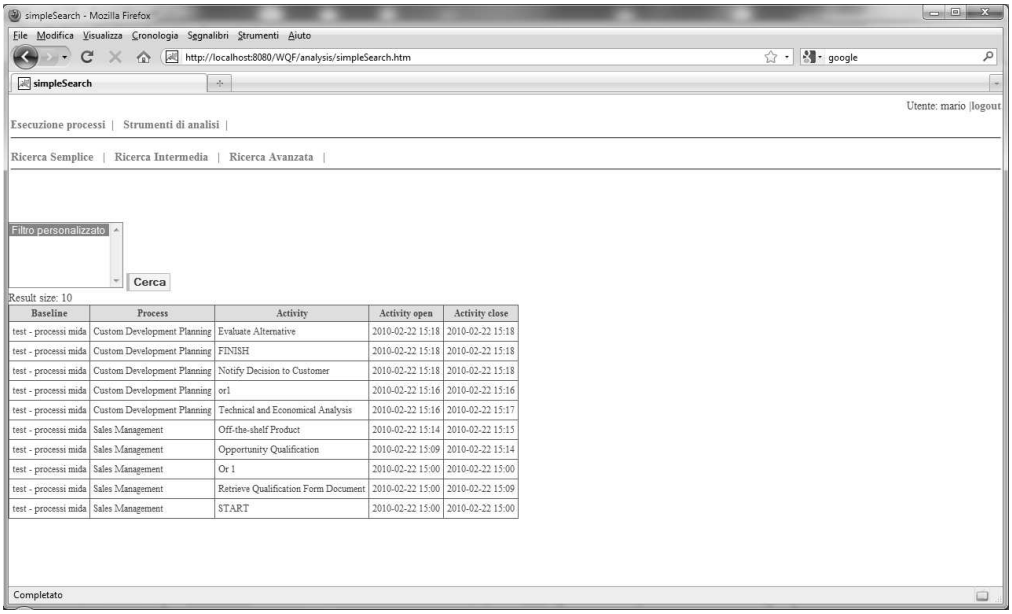


**Figura 14.9.** Ricerca semplice: utilizzo di un filtro pre-caricato.

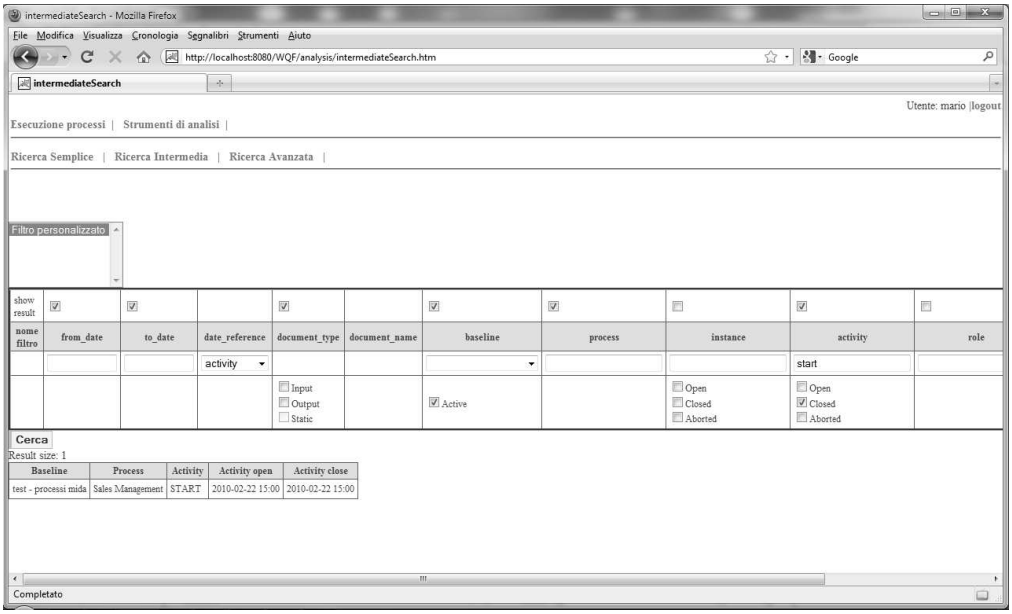


**Figura 14.10.** Ricerca intermedia: personalizzazione dei criteri di ricerca forniti da un filtro pre-caricato.

gere/eliminare campi di ricerca, mentre è possibile personalizzare i campi da

visualizzare nei risultati. Nella figura 14.10 è rappresentata la pagina di ricerca intermedia in cui l'utente "mario" carica il filtro di ricerca "Filtro personalizzato" descritto nella sezione precedente e lo modifica per far si che le attività cercate siano solo quelle che si chiamano "start".

### 14.2.4 Ricerca avanzata

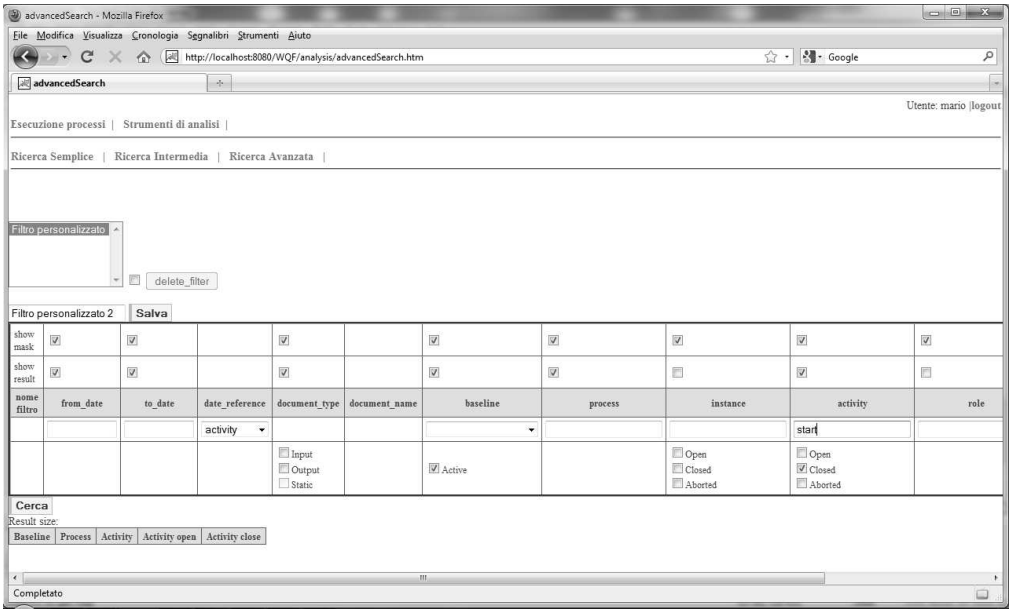


**Figura 14.11.** Ricerca avanzata: modifica di un filtro esistente e salvataggio di quello nuovo.

La pagina di `Ricerca avanzata` (vedi figura 14.11) permette il caricamento di filtri precedentemente creati, la loro modifica ed il salvataggio di nuovi filtri. È inoltre possibile eliminare i filtri creati da se stessi o quelli creati dall'amministratore appositamente per l'utente che si è autenticato. In particolare, la figura 14.11 mostra la modifica di "Fitro personalizzato" da parte dell'utente "mario" e il salvataggio dei nuovi criteri di ricerca come "Filtro personalizzato 2".

# Parte III

# Appendice

# Appendice A

# Sorgenti applicazione di prova

## A.1 Albero delle directory

In questa sezione viene riportata la struttura delle directory dell'applicazione `security_project` (vedi figura A.1) spiegando il significato di ciascuna cartella ed il relativo contenuto.

**.settings** : contiene le impostazioni dell'IDE Eclipse.

**build** : contiene i files compilati organizzati in package. Nel caso specifico di `security_project` contiene le classi `welcomeController.class` e `Permessi.class` nel package `myPack`.

**src** : contiene i files sorgenti organizzati in package. Nel caso specifico di `security_project` contiene le classi `welcomeController.java` e `Permessi.java` nel package `myPack`.

**WebContent** :

- **META-INF**: è una cartella specifica delle applicazione *Java-based*.
- **WEB-INF**: è il cuore di un'applicazione web. Solo la directory WEB-INF risulta inaccessibile all'utente che usa il browser cercando di raggiungere una pagina direttamente tramite l'indirizzo URL; per questo motivo verranno posizionati qui tutti i contenuti che si vogliono nascondere a coloro che non passano attraverso l'autenticazione. La cartella contiene, inoltre, i files di configurazione (quelli con estensione `.xml`) che vengono utilizzati da Tomcat.
  - **jsp**: come facilmente intuibile dal nome, contiene le pagine JSP.
  - **lib**: contiene le librerie necessarie al corretto funzionamento dell'applicazione (per esempio quelle di Tomcat, quelle di Spring Security, ecc.).
  - **tags**: contiene i tag personalizzati.

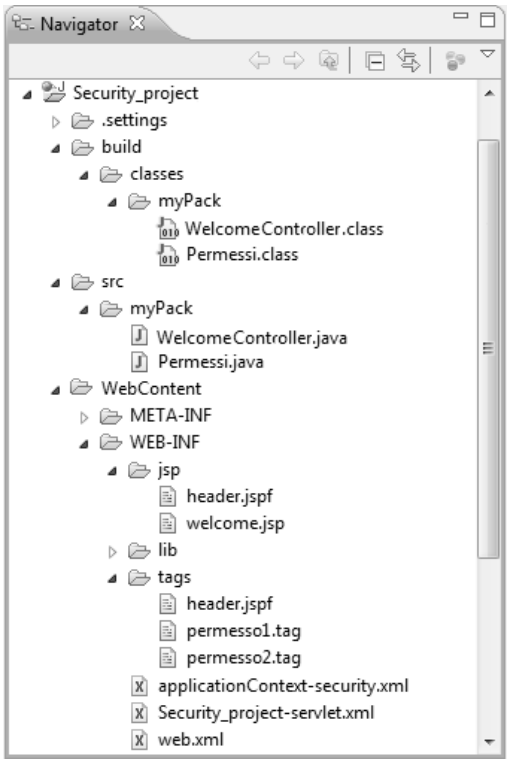


**Figura A.1.** Visione d'insieme delle directory dell'applicazione `security_project`.

## A.2 Codice sorgente

### A.2.1 `web.xml`

Il file è contenuto nella cartella `WEB-INF`.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance"
3      xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="
           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
4      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
5      id="WebApp_ID" version="2.5">
6      <display-name>Security_project</display-name>
7
8      <context-param>
9          <param-name>contextConfigLocation</param-name>
10         <param-value>
11             /WEB-INF/applicationContext-security.xml
12         </param-value>
```

```
13      </context-param>
14      <!--
15          - Loads the root application context of this web app
                at startup. - The application context is then
               available via WebApplicationContextUtils.
               getWebApplicationContext(servletContext). -->
16      <listener>
17          <listener-class>org.springframework.web.context.
               ContextLoaderListener</listener-class>
18      </listener>
19
20      <filter>
21          <filter-name>springSecurityFilterChain</filter-name>
22          <filter-class>org.springframework.web.filter.
               DelegatingFilterProxy</filter-class>
23      </filter>
24
25      <filter-mapping>
26          <filter-name>springSecurityFilterChain</filter-name>
27          <url-pattern>/*</url-pattern>
28      </filter-mapping>
29
30      <welcome-file-list>
31          <welcome-file>index.html</welcome-file>
32          <welcome-file>index.htm</welcome-file>
33          <welcome-file>index.jsp</welcome-file>
34          <welcome-file>default.html</welcome-file>
35          <welcome-file>default.htm</welcome-file>
36          <welcome-file>default.jsp</welcome-file>
37      </welcome-file-list>
38
39      <servlet>
40          <servlet-name>Security_project</servlet-name>
41          <servlet-class>org.springframework.web.servlet.
               DispatcherServlet</servlet-class>
42          <load-on-startup>1</load-on-startup>
43      </servlet>
44
45      <servlet-mapping>
46          <servlet-name>Security_project</servlet-name>
47          <url-pattern>*.htm</url-pattern>
48      </servlet-mapping>
49  </web-app>
```

### A.2.2 applicationContext-security.xml

Il file è contenuto nella cartella WEB-INF.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <beans:beans xmlns="http://www.springframework.org/schema/
       security"
4  xmlns:beans="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/
       schema/beans http://www.springframework.org/schema/beans
       /spring-beans-2.0.xsd http://www.springframework.org/
       schema/security http://www.springframework.org/schema/
       security/spring-security-2.0.4.xsd">
5
6      <http auto-config='true'>
7          <intercept-url pattern="/login.jsp*" filters="none"
               />
8          <intercept-url pattern="/**"  access="ROLE_USER,
               ROLE_PROGETTISTA,ROLE_ADMIN,ROLE_UNIVERSO"
               requires-channel="http" />
9          <form-login default-target-url="/welcome.htm" always
               -use-default-target="true" />
10         <logout logout-success-url="/index.htm"/><logout />
11     </http>
12
13     <!-- ********** IN-MEMORY AUTHENTICATION ********** -->
14     <authentication-provider>
15         <user-service>
16             <user name="mario" password="rossi" authorities=
                   "ROLE_ADMIN" />
17             <user name="giuseppe" password="verdi"
                   authorities="ROLE_USER" />
18             <user name="eva" password="ca" authorities="
                   ROLE_PROGETTISTA" />
19             <user name="dio" password="omni" authorities="
                   ROLE_UNIVERSO" />
20             <user name="sadmin" password="sadmin"
                   authorities="ROLE_USER,ROLE_UNIVERSO,
                   ROLE_PROGETTISTA,ROLE_ADMIN" />
21         </user-service>
22     </authentication-provider>
23 </beans:beans>
```

### A.2.3 Security_project-servlet.xml

Il file è contenuto nella cartella WEB-INF.

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/
   beans http://www.springframework.org/schema/beans/spring
   -beans-2.0.xsd">

    <bean name="/welcome.htm" class="myPack.
       welcomeController" />
    <bean id="viewResolver" class="org.springframework.web.
       servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>
</beans>
```

### A.2.4 welcome.jsp

Il file è contenuto nella cartella jsp.

```
<%@ include file="header.jspf" %>
<%@ page import="org.springframework.security.context.
   SecurityContextHolder" %>
<%@ page import="org.springframework.security.Authentication
   " %>
<%@ page import="org.springframework.security.
   GrantedAuthority" %>
<%@ page import="org.springframework.security.adapters.
   AuthByAdapter" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
   EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
           charset=ISO-8859-1">
        <title>Security_project welcome page</title>
    </head>
    <body>
        Benvenuto nel sistema!

        <authz:authorize ifAnyGranted="ROLE_ADMIN,
           ROLE_UNIVERSO" >
        <h3> scritta riservata a chi ha permesso universo o
           admin </h3>
```

```
18          </authz:authorize>
19          <%
20          Authentication auth = SecurityContextHolder.
               getContext().getAuthentication();
21          if (auth != null) { %>
22              <b><br>Sei autenticato con i seguenti ruoli:<br>
                   </b>
23              <%
24              GrantedAuthority[] granted = auth.getAuthorities
                   ();
25              for (int i = 0; i < granted.length; i++) { %>
26                  <%= granted[i].toString() %> <br>
27                  <%          }}%>
28          <b><br>permessi:<br></b>
29          permesso 1: <c:out value="${miopermesso.p1}" /><br>
30          permesso 2: <c:out value="${miopermesso.p2}" /><br>
31
32          <tags:permesso1 perm_value="${miopermesso.p1}"/><br>
33          <tags:permesso2 perm_value="${miopermesso.p2}"/><br>
34          <br>
35          <a href="<c:url value="/j_spring_security_logout" />
               ">logout</a>
36      </body>
37  </html>
```

### A.2.5 welcomeController.java

Il file è contenuto nella cartella src/myPack.

```
1  package myPack;
2
3  import javax.servlet.http.HttpServletRequest;
4  import javax.servlet.http.HttpServletResponse;
5
6  import org.springframework.security.Authentication;
7  import org.springframework.security.context.
       SecurityContextHolder;
8  import org.springframework.web.servlet.ModelAndView;
9  import org.springframework.web.servlet.mvc.Controller;
10
11 public class welcomeController implements  Controller {
12
13     public ModelAndView handleRequest(HttpServletRequest
           request, HttpServletResponse response) throws
           Exception {
14          Authentication auth = SecurityContextHolder.
               getContext().getAuthentication();
15          Permessi perm = new Permessi();
16          if (auth.getName().equals("mario"))
```

```
17          {
18              perm = new Permessi(2,1);
19          }
20          if (auth.getName().equals("giuseppe"))
21          {
22              perm = new Permessi(1,2);
23          }
24          if (auth.getName().equals("eva"))
25          {
26              perm = new Permessi(2,0);
27          }
28          if (auth.getName().equals("dio"))
29          {
30              perm = new Permessi(2,2);
31          }
32          return new ModelAndView("welcome","miopermesso",perm
                );
33      }
34  }
```

### A.2.6 Permessi.java

Il file è contenuto nella cartella src_myPack.

```
1  package myPack;
2
3  public class Permessi {
4
5      private int p1;
6      private int p2;
7
8      public Permessi() {
9          super();
10     }
11
12     public Permessi(int p1, int p2) {
13         super();
14         this.p1 = p1;
15         this.p2 = p2;
16     }
17
18     public int getP1() {
19         return p1;
20     }
21
22     public void setP1(int p1) {
23         this.p1 = p1;
24     }
25
```

```
26      public int getP2() {
27          return p2;
28      }
29
30      public void setP2(int p2) {
31          this.p2 = p2;
32      }
33  }
```

### A.2.7 header.jspf

Il file è contenuto nella cartella jsp.

```
1  <%@ page session="false"%>
2  <%@ page contentType="text/html"%>
3  <%@ page pageEncoding="ISO-8859-1"%>
4  <%@ page language="java" contentType="text/html"%>
5
6  <%@ taglib prefix="tags" tagdir="/WEB-INF/tags" %>
7
8  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c
       " %>
9  <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="
       fmt" %>
10 <%@ taglib uri="http://java.sun.com/jsp/jstl/functions"
       prefix="fn" %>
11 <%@ taglib uri="http://www.springframework.org/tags" prefix=
       "spring" %>
12 <%@ taglib uri="http://www.springframework.org/tags/form"
       prefix="form" %>
13 <%@ taglib prefix="authz" uri="http://www.springframework.
       org/security/tags"%>
```

# Bibliografia


[1] M. Bertocco, P. Callegaro, D. De Antoni Migliorati, *Ingegneria della Qualità*. De Agostini Scuola, 1st ed., 2006. ISBN: 978-88-825-172942.

[2] F. Rosso, M. Schimd, "Percorso di certificazione di Mida Solutions." tesina per il Corso di Ingegneria della Qualità, Facoltà di Ingegneria, Università degli Studi di Padova, 2009.

[3] A. Rampin, G. Lorenzin, M. Pighin, "Modelli di workflow management e basi di dati per la gestione di un'azienda in conformità alle norme ISO 9001." tesi di Laurea Triennale, Corso di Laurea Triennale in Ingegneria Informatica, Università degli Studi di Padova, 2009.

[4] F. Pezzutto, F. Tisiot, "Norme ISO/IEC TR 15504, Information Technology - Software process assessment." tesina per il Corso di Ingegneria della Qualità, Facoltà di Ingegneria, Università degli Studi di Padova, 2008.

[5] A. Martino, "Il Project Management." tratto dal sito dell'Autorità per l'Informatica nella Pubblica Amministrazione, 2009. www.aipa.it.

[6] R. S. Pressman, *Principi di Ingegneria del software*. McGraw-Hill, 5th ed., 2008. ISBN: 978-88-386-64182.

[7] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elementi per il riuso di software a oggetti*. Addison-Wesley, 1st italian ed., 2008. ISBN: 978-88-7192-150-1.

[8] "Spring Framework 2.5 Reference Manual." Online available, http://static.springsource.org/spring/docs/2.5.x/reference/, 2008.

[9] "Spring Framework 2.5 API (Javadoc)." Online available, http://static.springsource.org/spring/docs/2.5.x/api/index.html, 2008.

[10] S. Ladd, D. Davison, S. Devijver, C. Yates, *Expert Spring MVC and Web Flow*. Apress, 1st ed., 2006. ISBN: 978-1-59059-584-8.

[11] T. Risberg, R. Evans, P. Tung, "Introduction to Spring MVC, Developing a Spring Framework MVC application step-by-step." documentazione elettronica di Spring Framework, 2008. directory: /docs/MVC-step-by-step/pdf, file: spring-mvc-step-by-step.pdf.

[12] C. S. Horstmann, *Concetti di informatica e fondamenti di JAVA 2*. Apogeo, 2nd ed., 2002. ISBN: 978-8-85032-024-0.

[13] V. Chopra, S. Li, J. Genender, *Professional Apache Tomcat 6*. Wiley Publishing, Inc., 1st ed., 2007. ISBN: 978-0-471-75361-2.

[14] D. Berlin, G. Rooney, *Practical Subversion*. Apress, 2nd ed., 2006. ISBN: 1-59059-753-2.

[15] B. Collins-Sussman, B. W. Fitzpatrick and C. M. Pilato, "Version Control with Subversion." documentazione online, 2007.

[16] T. Oetiker, "The not so short introduction to LaTeX $2_\varepsilon$." documentazione online di Latex, 1999.

# Elenco delle tabelle

# Elenco delle figure

# Ringraziamenti

Dopo cinque anni (e qualche mese) sono, finalmente, giunto alla conclusione del mio percorso di studi universitari. A questo punto desidero ringraziare, ed esprimere loro la mia riconoscenza, tutte le persone che, in modi diversi, mi sono state vicine e hanno permesso ed incoraggiato sia i miei studi che la realizzazione e stesura di questa tesi.

Innanzitutto desidero ringraziare il *Prof. Matteo Bertocco* per la fiducia fin da subito dimostratami nell'avermi assegnato questo argomento di tesi e per avermi seguito durante lo svolgimento dello stesso con consigli e confronti che mi hanno aiutato ad intraprendere, ogni volta, le scelte più appropriate. Lo ringrazio, in particolare, per la continua disponibilità e prontezza nei charimenti e suggerimenti, per la rilettura critica di tutti i capitoli della tesi e per avermi guidato con i suoi suggerimenti durante la conclusione di questo percorso formativo. Un ringraziamento va anche all'*Ing. Mauro Franchin*, correlatore e rappresentante di Mida Solutions, per avermi dato la possibilità di svolgere un lavoro diverso dall'ordinario.

Grazie al mio compagno di corso *Andrea Molinaroli* per aver accettato di collaborare con me nel progetto, fornendo un contributo essenziale alla buona riuscita dello stesso, per aver messo a disposizione i locali del suo appartamento e per avermi tenuto sempre attivo con sostanziose dosi di caffè.

Ringrazio la mia famiglia e gli amici che mi sono stati molto vicini in tutti questi anni "da studente" e che, oltre ad avermi sempre "supportato", mi hanno, più di tutto, "sopportato". Il mio primo pensiero, ovviamente, va ai miei *genitori*, a cui dedico questo lavoro di tesi: senza il loro aiuto non avrei mai raggiunto questa meta. Sono davvero grato per tutto il sostengo economico, ma, più di ogni altra cosa, per quell'aiuto tacito o esplicito che è venuto dal loro cuore, per tutte le volte che mi hanno incoraggiato vedendomi preso dai libri, da un esame e da questa tesi. Mi auguro che tutti i sacrifici spesi siano in questo modo, almeno in parte, ripagati. Ringrazio *Giovanna*, che con estrema pazienza ha sopportato i miei sbalzi di umore e le mie paranoie quando, sotto stress, mi sfogavo in modo particolare con lei. Se ho raggiunto questo traguardo lo devo anche alla sua continua presenza, per avermi fatto capire che potevo farcela, incoraggiandomi a "non mollare mai" (ti devo delle vacanze!). Grazie, inoltre, ai *nonni* (anche quelli che non ci sono più), agli *zii* e ai *cugini*. Grazie anche a *Giulia*, *Giorgia*,

*Sandro* e *Serenella*, per me una seconda famiglia.

Come non ringraziare tutti gli Amici dell'Università ed in modo particolare i miei "Compagni di corso, di Progetti universitari e di Spritz" con i quali ho condiviso più da vicino questi ultimi due anni di intenso studio (ma anche di piacevoli svaghi). Loro, più di tutti, possono comprendere il mio grado di soddisfazione. Grazie, quindi, a *Patric*, *Christian*, *Alex* (fedele vicino di posto allo stadio), *Diego*, *Alberto* (per le scampagnate sul Grappa), *Daniele*, *Luca* e tutti gli altri conosciuti in questi cinque anni.

Per ultime, ma non certo per importanza, desidero ringraziare *Damaris* e *Gabriella*, le mie insegnanti delle elementari. Le ho già ringraziate al conseguimento della Laurea Triennale, ma voglio ripetermi in quanto ritengo indispensabile il loro contributo al mio percorso formativo giunto, ormai, a conclusione.

Aprile 2010

*Enrico*