

Capitolo 1: I sistemi informativi

1. Introduzione ai sistemi informativi

Definizione di sistema informativo

Il sistema informativo (SI) di un'azienda è l'insieme di:

- Le informazioni che l'azienda usa, produce e trasforma durante l'esecuzione dei processi aziendali.
- Le modalità in cui le informazioni sono gestite (attività).
- Le risorse, sia umane, sia tecnologiche, coinvolte.

Il sistema informativo si occupa quindi di gestire la risorsa informazione per una o più organizzazioni (o aziende), secondo determinate regole aziendali (dette *business rules*) e utilizzando determinate tecnologie.

Sistema informatico

Il sistema informativo non va confuso con il *sistema informatico*: questo termine infatti indica solamente la porzione di sistema informativo che fa uso di tecnologie informatiche e di automazione.

Possiamo quindi dire che un sistema informatico è un sistema che usa le tecnologie dell'ICT per elaborare, archiviare e scambiare informazioni. Un sistema informativo basato sull'ICT si dice anche *sistema informativo informatizzato*.

Complessità del sistema informativo

Un sistema informativo in una organizzazione è un sistema complesso. Si hanno infatti:

- Più sorgenti informative;
- Più basi di dati;
- Collegamenti in rete con reti locali e geografiche;
- Diversi processi interconnessi tra loro.

Esistono inoltre dei *sistemi informativi cooperativi*, che cioè riguardano più organizzazioni.

2. Risorse e processi

Il sistema informativo è strettamente correlato all'organizzazione per la quale è stato sviluppato. Alcuni concetti fondamentali dell'organizzazione, che si riflettono sul SI stesso, sono:

- Le risorse.
- I processi.

Le risorse

Le risorse sono rappresentate dalle persone (risorse umane), dalle tecnologie (risorse tecnologiche), ma anche dalle informazioni che vengono usate dall'organizzazione per il proprio funzionamento.

Una risorsa è quindi un "qualcosa" (persona, applicazione software, ...) che esegue un certo lavoro all'interno dell'organizzazione in analisi.

I processi (di business)

Un processo (di business) è l'insieme di attività che l'organizzazione nel suo complesso svolge:

- per gestire il ciclo di vita di una risorsa o di un gruppo omogeneo di risorse.
- per raggiungere un risultato definito e misurabile (prodotto/servizio).

I processi per operare hanno bisogno di informazioni. Le attività che costituiscono un processo sono generalmente svolte da più persone. Tali attività inoltre devono essere legate tra loro per mezzo del trasferimento di dati o da una relazione di sequenzialità.

È bene notare che non sempre gli individui che svolgono le singole attività sono consapevoli del fatto che l'attività stessa sia parte di un processo più ampio, e comunque la visione dei singoli individui non è mai globale: essi sono in grado al più di vedere le attività direttamente collegate a quelle che svolgono.

In sintesi, il processo di business è "ciò che si vuole che accada" all'interno dell'organizzazione.

3. Il modello di Anthony

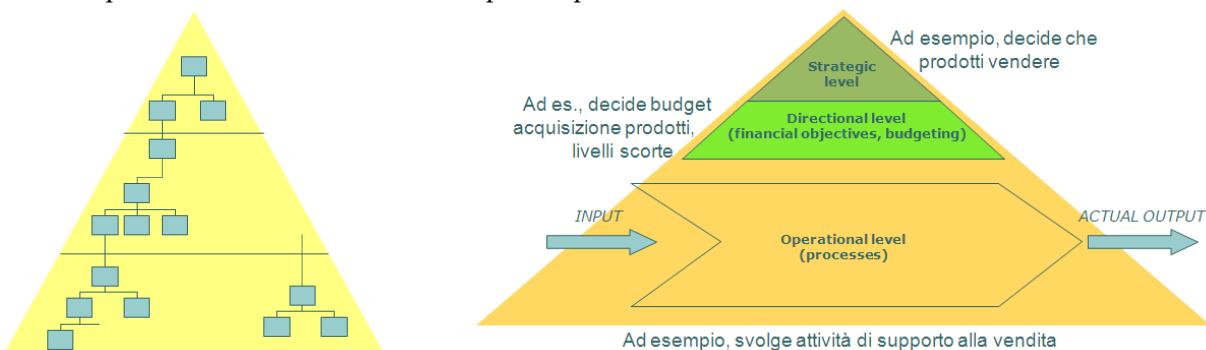
Cos'è la piramide di Anthony

La piramide di Anthony è un modello gerarchico di comportamento organizzativo, che ci consente di classificare i ruoli all'interno di un'organizzazione.

Essa, come una comune "piramide sociale", si articola su tre livelli di decrescente importanza:

1. Al più alto livello della piramide si trova il **livello strategico**, che si occupa di decidere gestionalmente le sorti dell'azienda.
2. Al secondo livello si trova il **livello tattico** o **direzionale (directional level)** che si occupa di organizzare le decisioni del livello sovrastante e di trasmetterle al livello operativo
3. Al terzo livello si trova il **livello operativo**, che progetta, crea e vende i prodotti e i servizi dell'azienda al grande pubblico.

Il principale motivo di questa disposizione è, secondo R. Anthony, l'autore dell'omonima piramide, che i tre diversi livelli hanno obiettivi gestionali o operativi di diversa lunghezza temporale e quindi di diversa importanza nel destino della società della quale fanno parte. Difatti il livello strategico ha obiettivi a lungo termine, quello tattico a medio termine e quello operativo a breve termine.



La struttura piramidale si riflette sulla struttura del sistema informativo.

Si noti inoltre che la piramide organizzativa è del tutto indipendente dalla dislocazione geografica degli individui che ricoprono le varie posizioni della piramide stessa.

Flussi informativi

All'interno della piramide di Anthony (ovvero all'interno dell'organizzazione) si verificano dei flussi informativi, ovvero degli scambi di informazioni. Tali scambi possono essere:

- ♦ **Orizzontali**

Si tratta di scambi tra persone che appartengono allo stesso livello della piramide di Anthony. Essi avvengono in seguito alla necessità di scambio di informazioni con altre persone o processi.

- ♦ **Verticali**

Si tratta di scambi tra persone che appartengono a livelli diversi della piramide di Anthony. Essi avvengono principalmente per segnalare (verso l'alto) e gestire (verso il basso) delle eccezioni.

Ad esempio, possiamo considerare lo scambio di informazioni che si ha nel caso in cui al livello operativo si dovesse riscontrare l'arrivo di merce non corrispondente a quella ordinata: il problema viene segnalato al diretto responsabile (livello tattico), che poi comunica la propria decisione al livello operativo.

4. Segmentazione dei sistemi informativi

La segmentazione

La segmentazione di un sistema informativo è la sua suddivisione in vari moduli. Come vedremo a breve, un SI può essere rappresentato mediante una *mappa*, che è il risultato della segmentazione dei processi gestionali. La mappa deve avere due caratteristiche fondamentali:

1. Deve essere un modello di alto livello, perciò deve offrire una visione sintetica di tutta l'azienda.
2. Deve spingersi a un dettaglio sufficiente ad individuare i singoli moduli del sistema informativo.

Modulo di un sistema informatico

Fino ad ora abbiamo usato il termine *modulo* in maniera piuttosto intuitiva. Cerchiamo però di capire meglio il significato di questo termine.

Un modulo di un sistema informativo è una funzionalità software che supporta una fase di un processo di business. Un modulo è omogeneo rispetto a:

1. L'implementazione software, ovvero:
 - a) Presentazione del software (aspetti grafici).
 - b) Logica applicativa.
 - c) Gestione dei dati.
2. Gli attori che lo utilizzano.
3. I dati gestiti.

Siccome si ha una sostanziale corrispondenza uno a uno tra le fasi dei processi di business e i moduli di un sistema informativo, un SI avrà tanti più moduli quanto più sono complessi i processi di business che vengono svolti all'interno dell'organizzazione alla quale il SI stesso è rivolto.

Moduli orizzontali e moduli verticali

I moduli si classificano in *verticali* ed *orizzontali*:

- ♦ **Moduli orizzontali**

Un modulo orizzontale è un modulo che non è strettamente legato ad un singolo settore, ma viene svolto esattamente allo stesso modo indipendentemente dal settore in analisi. Si dice quindi che questi moduli sono invarianti rispetto al settore di attività. Essi sono prevalentemente rivolti alle attività amministrative.

Ad esempio, il modulo che si occupa del pagamento degli stipendi è indipendente dal settore nel quale il dipendente lavora: il pagamento avviene sempre allo stesso modo (cambieranno solamente i dati, come ad esempio l'importo dello stipendio).

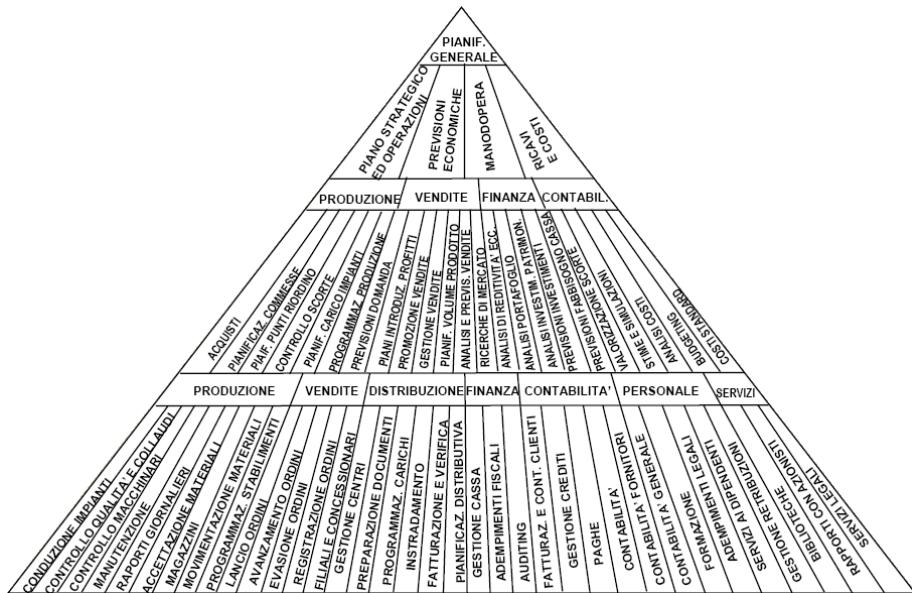
- ♦ **Moduli verticali**

I moduli verticali sono invece i moduli che supportano decisioni specifiche di un singolo settore (ad esempio, il settore delle telecomunicazioni è quello che riguarda il modulo relativo alla compagnia telefonica). Tali moduli rispecchiano le differenti transazioni operative di ogni settore industriale.



I moduli all'interno del modello di Anthony

Il modello di Anthoy consente di fatto di evidenziare i vari moduli "tagliando a fettine" i vari livelli della piramide:



Tale suddivisione però non consente di avere una visione globale sull'intera organizzazione, e per questo motivo sono stati introdotti anche altri modelli.

5. La catena del valore di Porter

Cos'è la catena del valore di Porter

La catena del valore è un modello che permette di descrivere la struttura di una organizzazione come un insieme limitato di processi. Tale modello mette in evidenza un modo di segmentazione del SI.

Questo modello è stato teorizzato da Michael Porter nel 1985. Secondo questo modello, un'organizzazione è vista come un insieme di 9 processi, di cui 5 primari e 4 di supporto:



Questa figura mette in evidenza che i processi di supporto corrispondono a moduli verticali del sistema informativo, mentre i processi primari corrispondono a moduli verticali.

Processi primari

I processi primari sono quelli che direttamente contribuiscono alla creazione dell'output (prodotti e servizi) di un'organizzazione. Nel modello in analisi, questi processi sono 5:

- ♦ **Logistica in entrata**
Comprende tutte quelle attività di gestione dei flussi di beni materiali all'interno dell'organizzazione.
- ♦ **Attività operative**
Sono le attività di produzione di beni e/o servizi.
- ♦ **Logistica in uscita**
Comprende quelle attività di gestione dei flussi di beni materiali all'esterno dell'organizzazione.
- ♦ **Marketing e vendite**
Sono le attività di promozione del prodotto o servizio nei mercati e di gestione del processo di vendita.
- ♦ **Assistenza al cliente e servizi**
Tutte quelle attività post-vendita che sono di supporto al cliente (ad es. l'assistenza tecnica).

I processi di supporto

I processi di supporto sono quelli che non contribuiscono direttamente alla creazione dell'output ma che sono necessari perché quest'ultimo sia prodotto e sono:

- ♦ **Approvvigionamenti**
L'insieme di tutte quelle attività preposte all'acquisto delle risorse necessarie alla produzione dell'output e al funzionamento dell'organizzazione.
- ♦ **Gestione delle risorse umane**
Sono le attività di ricerca, selezione, assunzione, addestramento, formazione, aggiornamento, sviluppo, mobilità, retribuzione, sistemi premianti, negoziazione sindacale e contrattuale, etc.
- ♦ **Sviluppo delle tecnologie**
Tutte quelle attività finalizzate al miglioramento del prodotto e dei processi. Queste attività vengono in genere identificate con il processo R&D (*Research and Development*).
- ♦ **Attività infrastrutturali**
Tutte le altre attività quali pianificazione, contabilità finanziaria, organizzazione, informatica, affari legali, direzione generale, etc.

Limiti del modello

1. Si adatta prevalentemente a grandi organizzazioni che trattano la produzione di beni. Quindi non si adatta bene alle piccole e medie imprese e alle organizzazioni che trattano prevalentemente servizi.
2. Sottostima l'importanza delle attività manageriali (planning, budgeting, decision-making ...). Nonostante questo, esso costituisce un valido spunto per l'analisi dei processi di un'organizzazione.

Capitolo 2: Gli ERP

1. Introduzione agli ERP

Le strategie di sourcing

Ogni azienda può procurarsi un sistema informativo in vari modi. Naturalmente, è molto raro che l'azienda debba partire completamente da zero nella costruzione un nuovo sistema informativo, anche nel caso in cui si dovesse trattare di un'azienda neonata: tale operazione avrebbe infatti dei costi decisamente troppo elevati.

Le modalità principali mediante le quali un'azienda può procurarsi un sistema informativo sono 2:

- ♦ ***La modalità In-source***

Consiste nel dotarsi di un sistema informativo "di proprietà". In particolare, si hanno due ulteriori scelte possibili:

- a) ***La scelta make***

Consiste nel "costruire" autonomamente il sistema informativo.

- b) ***La scelta buy***

Consiste nell'acquistare un sistema informativo dall'esterno. A questo punto, si hanno ulteriori possibilità:

1. ***One step shopping***

Si può scegliere di acquistare un unico sistema informativo, "in una sola volta" e da un unico fornitore. I sistemi informativi di questo tipo sono detti ERP.

2. ***Best of the Breed***

In questo caso, si sceglie di acquistare i diversi moduli del sistema informativo da diversi fornitori, cercando cioè la soluzione migliore per ognuno dei moduli. Naturalmente, il problema che si avrà sarà poi quello di dover integrare tra loro i moduli del sistema informativo.

- ♦ ***La modalità out-source***

L'altra possibilità è quella di usare un sistema informativo semplicemente come servizio, messo a disposizione da un'azienda esterna. Si parla in questo caso di SaaS.

Che cos'è un ERP

L'acronimo ERP significa *Enterprise Resource Planning* (*pianificazione delle risorse d'impresa*). Questo acronimo è stato creato all'inizio degli anni '90 da Gartner Group.

Gli ERP sono i package applicativi più diffusi nel mondo aziendale. Un ERP è un sistema informativo di tipo software, costruito al fine di automatizzare e standardizzare tutti i processi rilevanti per un'azienda (dalle vendite ai processi finanziari, ...). Esso è quindi costituito da un insieme di moduli orizzontali e verticali, che supportano l'intera gamma dei processi di un'azienda. I moduli verticali (o *suite settoriali*) comprendono i moduli specifici di un settore industriale. L'effettiva completezza delle suite settoriali dipende dal settore (è massima nel settore manifatturiero, mentre è più limitata in settori come la pubblica amministrazione o le banche).

Il suo obiettivo principale è quello di integrare tra loro tutte le informazioni dell'organizzazione e condividerle con i propri partner (ad esempio, mediante portali web).

I moderni sistemi di ERP coprono tutte le aree che possono essere automatizzate e/o monitorate all'interno di un'azienda, permettendo così agli utilizzatori di operare in un contesto uniforme ed integrato, indipendentemente dall'area applicativa. Dai primi anni del 2000, i maggiori vendor di soluzioni ERP hanno iniziato a dar vita a sistemi specializzati in uno specifico settore (destinati cioè ad organizzazioni di un particolare tipo).

Solitamente gli ERP offrono i migliori processi e le migliori implementazioni di dati. Di conseguenza, la scelta di un ERP anziché un altro non è più un elemento che consente di influenzare la competitività dell'azienda. L'ERP più importante e più diffuso è oggi SAP, prodotto dall'omonima azienda.

Da un punto di vista tecnologico, l'ERP è costituito da opportuni archivi per i dati e da un insieme di moduli (uno per ogni processo gestito dall'ERP).

Le piccole aziende e gli ERP

Gli ERP prodotti dalle grandi imprese multinazionali come SAP sono utilizzati soprattutto da imprese anch'esse di grandi dimensioni e, molto spesso, a carattere multinazionale. Le piccole e medie imprese invece usano sistemi informativi diversi, a seguito di varie ragioni:

1. Hanno capacità di spesa molto più ridotte, e i costi degli ERP standard sono spesso molto elevati;
2. Le piccole e medie imprese sono meno complesse, perché sono radicate in una sola località (non c'è bisogno di gestire diverse lingue) e, essendo tipicamente costituite da un solo impianto, possono essere governate a vista.
3. I processi che esse gestiscono sono più semplici (ad esempio, le piccole imprese hanno un solo magazzino, e questo semplifica i processi di gestione delle materie prime).

Gli ERP rivolti alle piccole e medie imprese perciò sono di tipo diverso:

a) *Package orizzontali semplificati:*

Le piccole e medie imprese acquistano pacchetti contenenti solo i moduli orizzontali fondamentali (ad esempio, per la gestione delle paghe e della contabilità) che eventualmente integrano con moduli verticali proprietari.

b) *Nicchie e package superverticali*

Molte piccole imprese sono altamente specializzate in un settore molto specifico, che quindi non viene gestito dagli ERP standard, mentre altri vendor di minori dimensioni si occupano anche della realizzazione di package specifici per tali settori.

c) *Versioni down-sized e prepopolate*

Molti vendor offrono soluzioni ridotte degli ERP standard, ad un prezzo inferiore, semplificate e con un numero inferiore di moduli.

d) *Modalità ASP*

Talvolta le piccole e medie imprese accedono tramite web a sistemi software preconfezionati, limitandosi poi a pagare l'uso di tale software.

Il paradigma ERP

Gli ERP rispecchiano una precisa concezione del sistema informativo aziendale, con 3 caratteristiche distintive che formano il paradigma funzionale detto *paradigma ERP*:

- ♦ *L'unicità dell'informazione*

L'informazione deve essere unica, perciò tutti i moduli del sistema condividono uno e un solo valore per una data informazione. Tale unicità avviene mediante l'uso di un'unica base di dati, che memorizza dati condivisi e sulla quale operano tutti i vari moduli.

Tale soluzione consente di:

1. Sincronizzare i dati ed evitare ridondanza e incongruenze.
2. Garantire la tracciabilità degli aggiornamenti. La tracciabilità è infatti un requisito di qualità della base di dati, che richiede di registrare i dati di ogni aggiornamento.
3. Integrare l'informazione direzionale, che risulta così anch'essa unica.

- ♦ *L'estensione e la modularità funzionale*

La suite ERP è costituita da moduli autosufficienti tra loro e ogni azienda può scegliere una strategia coerente con la propria realtà. L'azienda quindi può scegliere quali moduli utilizzare e, come abbiamo già visto analizzando le strategie di sourcing, può anche scegliere se integrare tra loro moduli di ERP diversi (strategia best of the breed).

Alla modularità si può associare anche la multi-nazionalità: un modulo funziona in diverse nazioni, con lingue diverse e notazioni diverse (ad esempio, l'uso della virgola o del punto come separatore per i decimali).

- ♦ *La prescrittività*

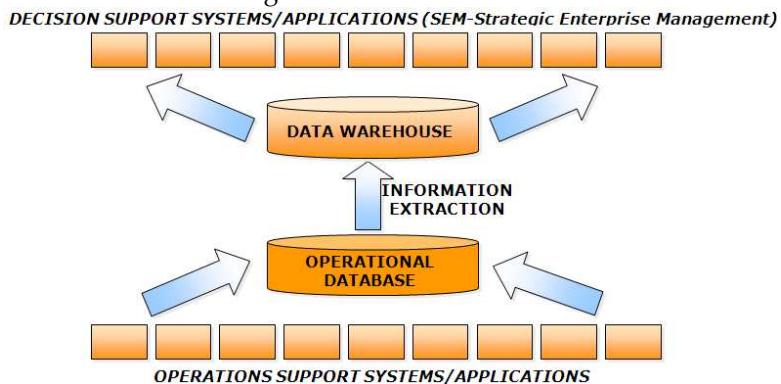
La prescrittività è la formazione dei processi gestionali derivante dal modello funzionale incorporato nel software. In altri termini, l'ERP definisce i processi secondo quelle che abbiamo già definito come le *best practice*, e sulla base di esse definisce delle regole (delle norme) che devono essere seguite. L'azienda risulta così costretta a conformare il proprio comportamento allo standard previsto dal sistema.

Ad esempio, si può definire la prescrizione "i materiali che entrano in azienda devono essere stati ordinati".

2. Come funzionano gli ERP?

La gestione dei dati

Come abbiamo più volte affermato, l'ERP è costituito da una serie di moduli, uno per ogni processo dell'organizzazione che l'ERP gestisce. Tuttavia, per garantire l'unicità dell'informazione, i dati risultano integrati, secondo lo schema mostrato di seguito:



Tale schema mette in evidenza la presenza di un'unica base di dati, alla quale accedono tutti i moduli relativi ai processi operativi. Tale database scambia poi informazioni con un data warehouse, il quale viene usato dai moduli relativi ai processi tattici e strategici (quelli dei primi 2 livelli della piramide di Anthony). La differenza fondamentale tra i dati all'interno del database e quelli nel data warehouse è data dal fatto che nel data warehouse troveremo soprattutto dati riguardanti proiezioni e previsioni per il futuro, mentre il database contiene solo informazioni riguardanti lo storico. Inoltre, il data warehouse viene aggiornato anche con altri dati provenienti dall'esterno.

Si noti inoltre che la centralizzazione della base di dati è di tipo logico: nella realtà è possibile che la base di dati sia fisicamente distribuita. La struttura logica centralizzata consente di ridurre la ridondanza e i problemi di coerenza dei dati.

I moduli

I moduli di un ERP si suddividono in due categorie:

- ♦ ***ERP core***

Sono i moduli principali, che riguardano i processi interni all'organizzazione stessa. Essi comprendono:

- a) **Moduli intersettoriali**

1. *Moduli istituzionali*

Si occupano dell'amministrazione dell'azienda e della gestione delle risorse umane.

2. *Moduli direzionali*

Ad esempio: la pianificazione strategica, la programmazione ed il controllo del budget, moduli di gestione progetti & investimenti, ...

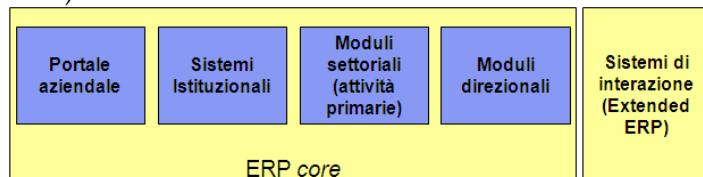
3. *Portale aziendale*

- b) **Moduli settoriali**

Sono moduli che supportano le attività primarie tipiche di un settore (per esempio, il ciclo di trasformazione di una azienda manifatturiera).

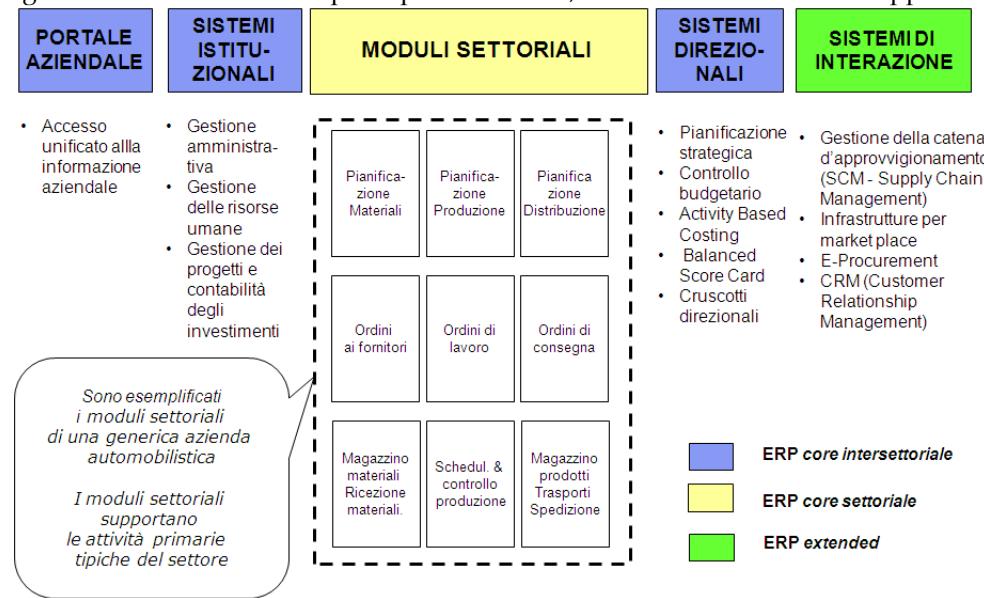
- ♦ ***Extended ERP***

Sono i moduli che riguardano l'interazione dell'organizzazione con il mondo esterno (ovvero le transazioni interaziendali).



Schema riassuntivo riguardante i moduli di un ERP

Lo schema seguente riassume i moduli principali di un ERP, classificandoli nel modo appena introdotto:



Esempio di modulo: Human Resources

Ogni processo si focalizza sulla gestione di una particolare risorsa per l'azienda in questione. Ogni risorsa ha un proprio ciclo di vita, nel quale le fasi principali sono:

1. Pianificazione
2. Acquisizione
3. Gestione
4. Conclusione

Ad esempio, se consideriamo il modulo che gestisce le risorse umane (ovvero il modulo corrispondente al processo HR – Human Resources), il ciclo di vita può essere riassunto nella seguente figura:



Ciò significa che il modulo HR dell'ERP dovrà definire le modalità mediante le quali gestire ciascuna di queste attività. Più concretamente, l'ERP metterà a disposizione delle schermate (interfacce) che, usando opportuni dati e facendo riferimento alle configurazioni selezionate dall'organizzazione (ad esempio, le leggi vigenti nel paese nel quale l'azienda opera) permettono di automatizzare l'attività stessa.

Ad esempio, per quanto riguarda la pianificazione di viaggi, verranno mostrate delle opportune finestre, nelle quali selezionare la destinazione, la data e l'orario di partenza, L'ERP proporrà poi alcune soluzioni (ad esempio, il volo, l'albergo, ...).

Configurazioni e dati configurabili

Un'altra importante caratteristica degli ERP, che è stata parzialmente introdotta mediante il precedente esempio, è data dalla possibilità di selezionare opportune configurazioni: nell'esempio si faceva riferimento alla scelta del paese, che consente implicitamente di adattarsi alle normative di legge vigenti in quello Stato (per esempio, in materia di pagamento stipendi).

Esistono inoltre dei dati *configurabili*, che possono essere inseriti in maniera facilitata dall'ERP stesso: ad esempio, la data non viene semplicemente digitata da tastiera, ma la si sceglie da un calendario, oppure la destinazione viene scelta mediante tendine che permettono di selezionare prima lo stato, poi la città,

Ciò che invece non cambia mai, indipendentemente dalla configurazione o dai dati configurabili, è la struttura dell'interfaccia.

3. La configurazione dell'ERP?

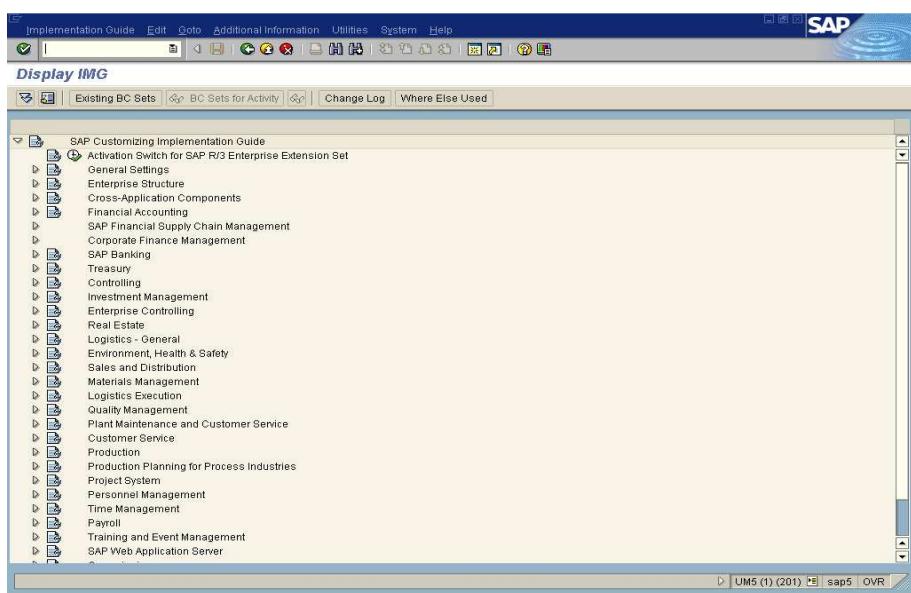
La configurazione

La configurazione di un ERP è il processo mediante il quale una singola organizzazione adatta il sistema software standard che ha acquistato alle esigenze del proprio business.

Come avviene la configurazione dell'ERP

All'atto della configurazione dell'ERP, l'organizzazione dovrà:

1. Selezionare quali sono i moduli di interesse e quali non lo sono, tipicamente mediante la scelta di una serie di opzioni all'interno di una finestra (cioè spuntando le voci di una checklist).
2. Effettuare un numero solitamente molto elevato di scelte di configurazione (ad esempio, il paese, ...). Si pensi ad esempio che SAP richiede oltre 8000 scelte di configurazione.
3. Selezionare la struttura dei dati (i canali di distribuzione, la suddivisione delle vendite, ...).
4. Riscrivere alcune parti di codice (anche se tale operazione è sconsigliata, perché potrebbe portare a problemi di incompatibilità all'atto dell'aggiornamento dell'ERP).
5. Scrivere nuove parti di codice, per scopi personalizzati.



4. Vantaggi e svantaggi degli ERP

Vantaggi

Vediamo quali sono i principali vantaggi derivanti dall'uso di un ERP:

1. Gli ERP fanno riferimento alle *best practises*: gli sviluppatori dei moduli dell'ERP interrogano un elevato numero di aziende, per individuare quali sono le modalità migliori per la gestione dei vari processi.
2. L'uso di un ERP comporta perciò un'ottimizzazione dei processi, che porta solitamente ad una riduzione dei costi di produzione.
3. Si ha una migliore gestione delle risorse, e quindi:
 - a) Si riducono i costi della gestione del magazzino, dell'acquisizione di materie prime, ...
 - b) Si riducono i tempi di produzione.

Svantaggi

Si hanno però anche alcuni considerevoli svantaggi:

1. Gli ERP consentono una flessibilità molto ridotta nel modo in cui svolgere le attività.
2. L'uso dell'ERP ha un forte impatto sull'organizzazione dell'azienda: comporta cioè un cambiamento dei processi e l'introduzione di una forte pianificazione delle transizioni.
3. I costi di licenza sono generalmente molto elevati.

5. Un esempio di ERP: SAP

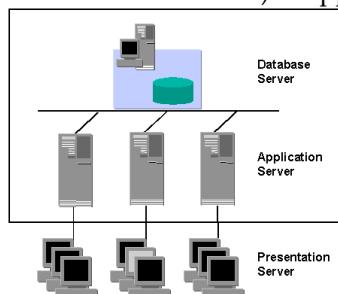
L'esempio SAP

Come abbiamo detto, SAP è l'ERP più usato al mondo. Si tratta di un sistema molto complesso, pensato soprattutto per le grandi aziende, sviluppato dall'omonima azienda tedesca nel 1972.

L'ERP SAP usava inizialmente database di tipo gerarchico, ma oggi utilizza basi di dati relazionali (ricordiamo che nel 1972 le basi di dati relazionali non erano ancora usate nella pratica).

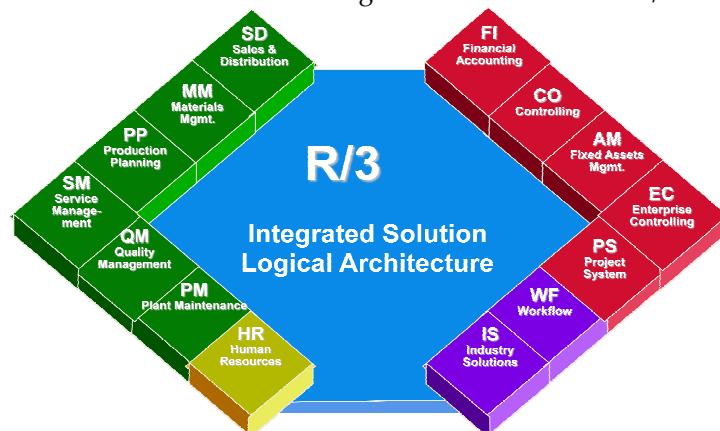
Schema di un sistema SAP R/3

La struttura di un sistema SAP R/3 (una delle versioni di SAP) è rappresentata dalla seguente figura:



Architettura logica e principali moduli di SAP

La figura seguente mette in evidenza l'architettura logica di un sistema SAP R/3 e i suoi moduli principali.



Capitolo 3: I CRM

1. Introduzione ai CRM

Che cosa sono i CRM?

Con CRM (Customer Relationship Management) si intende l'insieme delle funzionalità che servono per la gestione delle relazioni dell'azienda con i propri clienti. In particolare, un CRM è una suite di software per il supporto al ciclo vitale del cliente, dalle campagne di offerta alla vendita e all'assistenza post-vendita. I sistemi CRM supportano le transazioni con il cliente su molteplici canali (web, negozi, agenti e reti di vendita, call center).

Nei CRM viene abbastanza a mancare la parte di gestione dei processi: non è così fondamentale andare a ricercare le "best practice". Di conseguenza, spesso i CRM vengono costruiti da zero, a differenza invece degli ERP: in questo caso l'alternativa "make" or "buy" vede "in vantaggio" la scelta "make".

La centralità del cliente

Fino ad ora abbiamo visto infatti gli strumenti che servono per gestire un'azienda "chiusa in sé stessa", dapprima descrivendone la struttura interna (piramide di Anthony), e analizzando poi i sistemi ERP (che però adottano sempre un punto di vista interno). Rimane dunque da analizzare il punto di vista dei clienti. Tale punto di vista è quello assunto dai CRM, che risultano quindi orientati al cliente, e questo comporta notevoli investimenti in strutture d'assistenza e, soprattutto, in sistemi informatici: un CRM sarebbe infatti impossibile senza sistemi informatici, perché oggi l'interazione del cliente con l'azienda avviene sempre più frequentemente mediante sistemi on-line. Di conseguenza, non solo è cambiato il contatto con il cliente, ma si possono anche raccogliere i dati relativi all'interazione con il cliente, al fine di migliorare la risposta del sistema alle richieste future del cliente stesso.

ERP e CRM

Nelle imprese, i sistemi CRM sono complementari ai sistemi ERP. Come abbiamo già sottolineato infatti, mentre gli ERP informatizzano le attività interne, i CRM informatizzano le transazioni verso i clienti.

Di conseguenza, il CRM deve poi essere integrato con l'ERP; alcuni ERP sono stati estesi per includere le funzionalità dei CRM, ma in molti altri casi si è preferito creare dei sistemi informativi separati.

Cenni all'evoluzione storica dei CRM

- ♦ *Anni Ottanta: la nascita degli SFA*

Nei primi anni Ottanta sono nati gli SFA (*Sales Force Automation systems*), ovvero dei sistemi che consentivano ai venditori di interagire con i clienti in una maniera nuova: erano infatti i venditori ad andare dai clienti, e non viceversa. Di conseguenza, essi sono adoperati spesso in contesti nei quali gli operatori di vendita sono sparsi sul territorio.

Più nel dettaglio, gli SFA:

1. Provvedono alla comunicazione tra il venditore e la centrale operativa;
2. Programmano e controllano l'azione dei venditori;
3. Assistono i venditori nella messa a punto di piani di vendita o promozione di un prodotto;
4. Sussidiano la raccolta degli ordini dei clienti.

Per esempio un venditore potrebbe avvisare un cliente di una nuova promozione, non sapendo che il cliente aveva già parlato con un collega dell'assistenza per un problema su un altro prodotto. Non sarebbe questo il momento opportuno per proporre una nuova offerta. Tramite i sistemi SFA si evitano problemi di questo tipo, causati dalla non sincronia dei vari organi o reparti di un'azienda.

- ♦ *Anni Novanta: i call center*

Negli anni Novanta sono poi nati i call center e i numeri verdi, che consentono di realizzare servizi di prenotazione, attività di vendita via telefono, servizi dopo la vendita (help desk), ...

- ◆ **1995: il Web**

Nel 1995 il Web ha iniziato ad essere usato come sistema per le vendite. Il Web consente di dare informazioni (mediante la visualizzazione di cataloghi) e di effettuare dei veri e propri acquisti.

- ◆ **Ultimi anni Novanta**

Negli ultimi anni Novanta si è assistito ad un'integrazione di tutti i precedenti sistemi all'interno di un unico CRM.

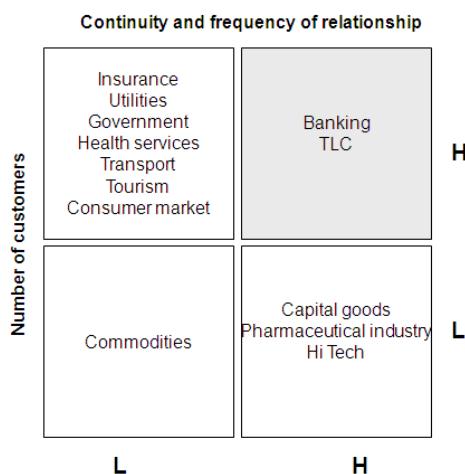
Di conseguenza, oggi il CRM mette a disposizione dell'utente diverse interfacce per l'acquisto: telefono, internet, venditori, Tutti i canali, anche se diversi tra loro, sono integrati su un'unica base di dati: indipendentemente dal canale sul quale un'operazione viene compiuta, i dati devono infatti essere aggiornati (storicamente, tale integrazione è stata un problema non facile da risolvere).

Quando è necessario usare un CRM?

A seconda dei diversi settori aziendali, si hanno differenze anche sostanziali nelle relazioni con i clienti. A seconda delle interazioni tra clienti e azienda, può essere più o meno utile che l'organizzazione stessa si doti di un CRM. Nel dettaglio, le caratteristiche che è interessante osservare sono:

- a) L'intensità delle relazioni tra l'azienda e i clienti, in termini di:
 1. Frequenza delle interazioni;
 2. Continuità delle interazioni;
 3. Lealtà dei clienti.
- b) Il numero di clienti dell'organizzazione.
- c) L'eventuale approccio multi-canale ai clienti. Diciamo che si ha un approccio multi-canale se l'azienda mette a disposizione dei clienti diversi modi per relazionarsi con essa (ad esempio, il web e un servizio di call center).

Per individuare i settori nei quali l'implementazione di un CRM risulta essere strategica, è possibile costruire la *matrice di posizionamento* (in inglese, *position matrix*), che tiene conto di due fattori fondamentali, ovvero la continuità e frequenza delle relazioni da un lato e il numero di clienti dall'altro.



il CRM risulta particolarmente utile quando ci troviamo nel quadrante in alto a destra, ovvero quando si hanno molto clienti, che interagiscono frequentemente e continuativamente con l'azienda. In misura inferiore, il CRM risulta utile quando ci si trova nel quadrante in alto a sinistra (cioè le relazioni sono meno frequenti e continue, ma il numero di clienti è molto alto).

Come messo in evidenza dal precedente schema, un settore in cui l'uso del CRM è particolarmente importante è il settore bancario.

2. Architettura di un CRM

Moduli di front-end e moduli di back-end

Un CRM è fondamentalmente costituito da un insieme di moduli, che vengono suddivisi in:

- ♦ ***Moduli di front-end***

Sono i moduli che costituiscono l'interfaccia verso il cliente. Ogni modulo consente di automatizzare un singolo canale di comunicazione con i clienti.

- ♦ ***Moduli di back-end***

Sono i moduli che si occupano di eseguire realmente le operazioni richieste dal cliente e di gestire i dati del cliente.

I diversi canali di comunicazione con il cliente (cioè i diversi moduli di front-end), anche se molto diversi tra loro, devono essere integrati su un'unica base di dati: indipendentemente dal canale sul quale un'operazione viene compiuta, si deve poter accedere a dati sempre aggiornati e si deve permettere che l'operazione venga registrata in modo da essere visibile anche da tutti gli altri canali. Storicamente, tale integrazione è stata un problema non facile da risolvere.

Suddivisione funzionale di un CRM

Il CRM però non si limita solamente ad eseguire le richieste del cliente: fornire un servizio infatti comporta la necessità di eseguire anche molte altre operazioni.

Per questo motivo, il CRM può anche essere suddiviso in maniera funzionale nelle 3 parti seguenti:

- ♦ ***CRM operativo (operational CRM)***

Il CRM operativo (o operazionale) è l'insieme dei moduli che informatizzano i canali attraverso i quali avvengono i contatti con il cliente.

In generale, il CRM mette a disposizione più di un canale di interazione con il cliente:

1. **Il canale "in presenza"**

Il cliente interagisce in presenza con dei rappresentanti dell'azienda. Si ha perciò una rete di negozi e/o filiali e/o di agenti porta a porta.

In questo caso, CRM interagisce più con il venditore che con il cliente, e perciò si parla di SFA (*Sistemi per le forze di vendita, Sales Force Automation*).

2. **Il canale "voce"**

Il cliente interagisce via telefono con una struttura di operatori, assistita da un sistema CRM integrato con apparati telefonici.

Si hanno quindi due componenti:

- a) La componente telefonica, formata da un centralino intelligente, dotato di dispositivi che automatizzano il dialogo con i clienti. I centralini possono essere molteplici, e può essere necessaria una complessa rete per la distribuzione delle chiamate.

- b) La componente informatica è costituita da un sistema interattivo, usato da un operatore per l'assistenza al cliente.

3. **Il canale "web"**

Il cliente interagisce attraverso un portale transattivo accessibile mediante il web da sistemi terminali (PC, telefono cellulare, ...). Questo canale è attivo 24 ore su 24.

4. **Il canale "corrispondenza"**

Il cliente comunica attraverso un testo scritto su supporto cartaceo, fax, SMS o posta elettronica. È quindi necessaria una rete di operatori che esamina le richieste agendo poi su sistemi opportuni. Si tratta di un canale lento, usato per piccoli volumi, e spesso complementare rispetto ad altri canali.

- CRM analitico (*analytic CRM*)

I moduli del CRM analitico sono tutti quei moduli che informatizzano la conoscenza sul cliente, usando tecnologie di business intelligence e una vasta famiglia di tecnologie di analisi dei dati. Questi moduli quindi si occupano dell'analisi del comportamento dei clienti.

In altri termini, il CRM analitico gestisce i profili e i comportamenti dei clienti, analizzandoli per individuare eventuali comportamenti anomali e/o gestire offerte commerciali personalizzate. Per fare ciò, interagisce con il CRM operativo, perché riceve informazioni da quest'ultimo, e perché usa il CRM operativo per la comunicazione con il cliente. Possiamo quindi dire che il CRM analitico e quello operazionale sono complementari.

Ad esempio, il CRM analitico può essere usato per selezionare i clienti ai quali rivolgere una certa offerta (sulla base del loro precedente utilizzo di un servizio – campagne di marketing personalizzate), e poi il call center provverà a chiamare i clienti interessati.

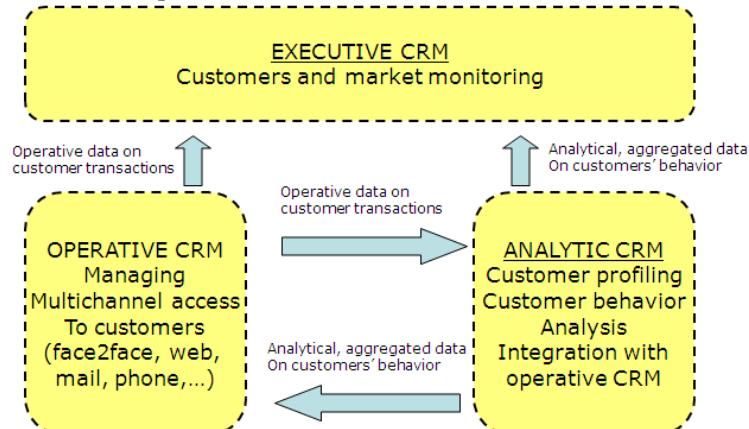
Un altro esempio di utilizzo del CRM analitico è relativo all'individuazione di comportamenti anomali da parte del cliente: proviamo a pensare al servizio bancomat. Se la frequenza di operazioni compiute mediante il bancomat in un certo periodo è molto superiore rispetto alla media, è possibile che il bancomat sia stato clonato, e quindi è opportuno segnalarlo al cliente.

- CRM esecutivo (*executive CRM*)

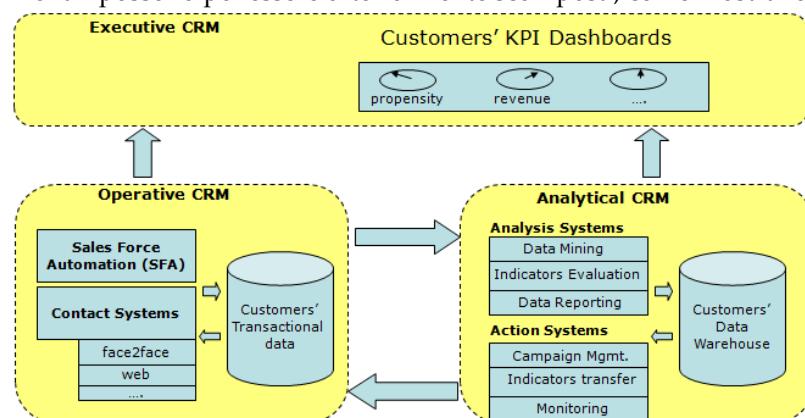
Il CRM esecutivo (o CRM direzionale) è un sistema di controllo, orientato al cliente. Esso monitora la definizione degli obiettivi, la loro assegnazione al management, e gestisce la misurazione dei risultati e le azioni correttive da intraprendere. Generalmente, il CRM direzionale si fonda su 3 principali classi di variabili:

1. La redditività economica del cliente;
2. Il livello di servizio alla clientela;
3. Il livello di soddisfazione della clientela.

La figura seguente mostra le varie parti del CRM e le relazioni tra esse:



I singoli blocchi funzionali possono poi essere ulteriormente scomposti, come mostra lo schema seguente:



Suddivisione del CRM analitico

Il CRM analitico può essere scomposto in due parti fondamentali:

- ♦ ***Analysis Systems***

Sono i moduli che si occupano di analizzare i dati del sistema. Comprendono:

- a) ***Moduli di data-mining***

Sono i moduli che cercano di individuare eventuali correlazioni tra i vari dati (ad esempio, l'importo dello stipendio di un cliente della banca ed il ritardo nel pagamento di una rata di un prestito).

Le tecniche che si usano sono quelle della correlazione, delle reti neurali, delle regole di associazione, degli algoritmi genetici,

- b) ***Indicators evaluation***

Sono i moduli che si occupano di definire quali sono gli indicatori delle performance che il sistema offre verso i clienti, e che gestiscono la valutazione automatica di questi stessi indicatori.

Per ogni indicatore, bisogna individuare caratteristiche come il nome, l'unità di misura, la metrica e la posizione all'interno della *gerarchia di valutazione* (cioè l'importanza dell'indicatore stesso).

- c) ***Data reporting***

È l'insieme dei moduli che mostrano all'utente finale i dati in maniera integrata e con un aspetto amichevole (ad esempio, attraverso l'uso di opportuni grafici).

La presentazione deve essere "centrata sull'utente", che è in questo caso il manager dell'azienda.

- ♦ ***Action Systems***

Sono i moduli che si occupano di mettere in atto delle azioni, sulla base delle analisi fatte dalla componente precedente. All'interno di questo insieme di moduli possiamo distinguere alcune sottocategorie:

- a) ***Campaign Management***

Sono i moduli che mirano ad individuare le strategie da mettere in atto per raggiungere nuovi clienti. Tali moduli forniscono quindi un appoggio per i processi che hanno lo scopo di conquistare nuove fette di mercato, come ad esempio le campagne pubblicitarie.

- b) ***Indicators transfer***

Sono i moduli che gestiscono l'interazione con il CRM operativo. In particolare, traducono l'interpretazione degli indicatori in azioni da compiere.

- c) ***Monitoring of critical variables***

Appartengono a questa categoria tutti i moduli che monitorano alcune variabili considerate particolarmente critiche (ad esempio, in una compagnia telefonica è una variabile critica quella che indica la gestione della rete telefonica stessa).

Tali moduli si occupano poi di segnalare eventuali valori anomali di queste variabili, per mezzo di messaggi in tempo reale (come SMS ed e-mail), oppure mediante dei pop-up.

Capitolo 4: Architetture ICT

1. Introduzione alle architetture ICT

La suddivisione in moduli

Come abbiamo già ampiamente visto, i sistemi informativi sono composti da moduli diversi. Tale caratteristica è legata a diverse motivazioni:

1. La necessità di segmentare il sistema stesso;
2. Tale divisione rende possibile far evolvere il sistema informativo in maniera incrementale.
3. L'uso di diversi moduli consente far inter-operare i sistemi informativi di aziende diverse.

Nascono così dei sistemi informativi cooperativi. Le diverse parti del sistema informativo (di tipo informatico) possono poi essere collegate mediante architetture diverse. Tali architetture sono le stesse che si hanno genericamente nei sistemi informatici.

Nell'ambito dei Sistemi Informativi, il termine *architettura* indica l'insieme delle scelte tecniche e organizzative che influiscono sullo sviluppo e sull'utilizzo delle risorse tecnologiche di tipo ICT.

2. Architetture centralizzate e distribuite

I sistemi informativi informatici possono essere centralizzati oppure distribuiti.

Sistemi informativi centralizzati

Un sistema informatico è detto centralizzato se i dati e le applicazioni risiedono in un unico nodo elaboratore. L'utente che deve accedere al sistema usa invece tipicamente un *terminale stupido*, cioè privo di capacità di elaborazione locale.

Da un punto di vista storico, l'architettura centralizzata è stata la prima ad essere utilizzata: essa è infatti stata introdotta negli anni '50, con la nascita dell'informatica moderna.

Sistemi informativi distribuiti

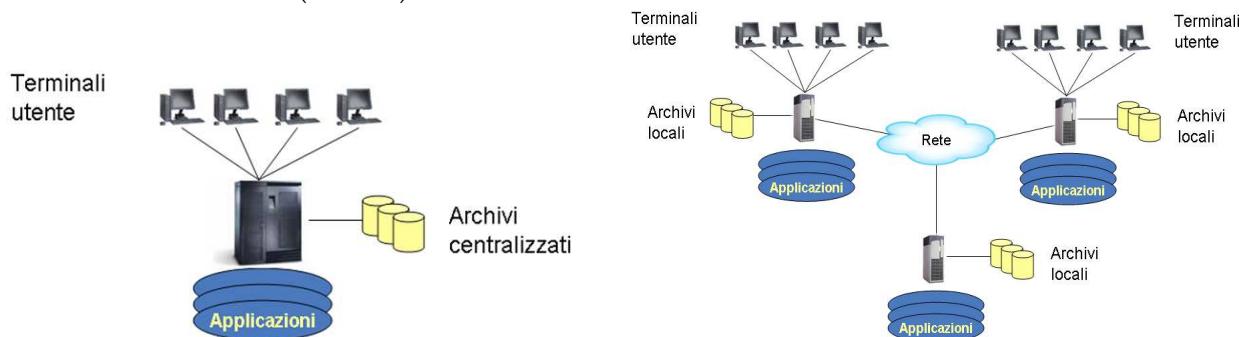
Si parla di sistema distribuito quando è verificata almeno una delle seguenti condizioni:

- a) Le applicazioni, tra loro cooperanti, risiedono su più nodi elaboratori (elaborazione distribuita).
- b) Il patrimonio informativo (che a livello logico è unitario) è ospitato su più nodi elaborativi (base di dati distribuita).

Un sistema distribuito è quindi costituito da un insieme di applicazioni logicamente indipendenti, che collaborano per il perseguitamento di obiettivi comuni, mediante un'infrastruttura di comunicazione hardware e software (rete).

L'architettura distribuita ha avuto sviluppo grazie all'evoluzione dei mainframe, alla nascita dei sistemi operativi time-sharing e allo sviluppo di tecnologie più economiche, e ha così sostituito l'architettura centralizzata. Tuttavia, nei primi anni Novanta il modello distribuito è stato sottoposto a forte critica a causa della sua maggior complessità progettuale e di gestione.

La figura seguente mette a confronto lo schema dell'architettura centralizzata (a sinistra) con quello di un'architettura distribuita (a destra).



3. Proprietà delle architetture e distinzione tier-layer

Proprietà delle architetture

Le varie architetture di un sistema informativo si differenziano tra loro per una serie di proprietà importanti, che ci consentono di effettuare dei confronti reciproci. Vediamo quali sono queste proprietà

- ♦ **Affidabilità**

L'affidabilità di un sistema è definita come la probabilità che il sistema sia operativo. Chiamiamo MTTF (*mean time to failure*) il tempo medio di funzionamento e MTTR (*mean time to repair*) il tempo medio di riparazione; in tal caso, vale la formula:

$$\text{Affidabilità} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

- ♦ **Scalabilità**

La proprietà di scalabilità di un sistema indica la capacità del sistema stesso di soddisfare richieste crescenti con aggiornamenti adeguati.

Diciamo quindi che un sistema non è scalabile quando il costo aggiuntivo da affrontare per soddisfare l'aumento delle richieste è eccessivo, oppure la crescita del traffico semplicemente non può avvenire.

Questa proprietà è importante perché a priori non è possibile stimare con esattezza il numero degli utenti che useranno il sistema, perciò è bene che quest'ultimo possa adeguarsi all'eventuale crescita di tale numero, senza la necessità di cambiare il sistema stesso.

- ♦ **Tolleranza ai guasti**

La tolleranza ai guasti è la capacità del sistema di funzionare anche in presenza di guasti di alcune delle sue componenti. Perché il sistema si tollerante ai guasti, si deve avere ridondanza delle sue componenti in modo tale da sostituirle l'una con l'altra quando ciò si rende necessario.

I livelli logici e fisici

Un sistema informativo complesso, che prevede la presenza di più sorgenti informative e basi di dati, è costituito da diversi processi interconnessi tra loro e necessita di collegamenti in rete (con reti locali e/o geografiche). Di conseguenza, si tratta solitamente di sistemi distribuiti.

Al fine di ottenere le proprietà descritte nel precedente paragrafo, il sistema informativo viene partizionato in livelli. In particolare, si distinguono

- ♦ **Livello logici (detti layer)**

La divisione in livelli logici corrisponde ad una divisione funzionale delle diverse parti del sistema informativo.

- ♦ **Livelli fisici (detti tier)**

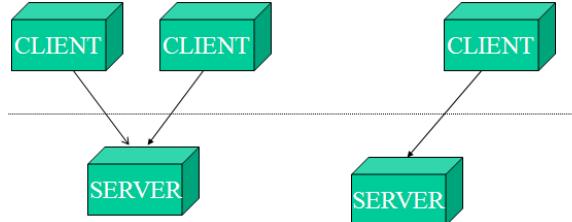
Avere diversi livelli fisici significa invece avere concretamente degli elaboratori (o insiemi di elaboratori) diversi tra loro, posti a livelli gerarchici differenti.

Come vedremo, la suddivisione in livelli logici e quella in livelli fisici non sempre coincide.

4. Architettura a livelli: architetture logiche

Architettura client server (2 livelli logici)

Questa architettura prevede che si separi l'interfaccia messa a disposizione degli utenti (client) dal servizio vero e proprio (server). Inizialmente i client erano specifici per una certa applicazione.



Tipicamente si possono avere più client che richiedono ad un server lo stesso servizio. Il tipico esempio è quello del DBMS.

Questa architettura può essere onerosa in termini di potenza di calcolo.

Architetture a 3 livelli logici

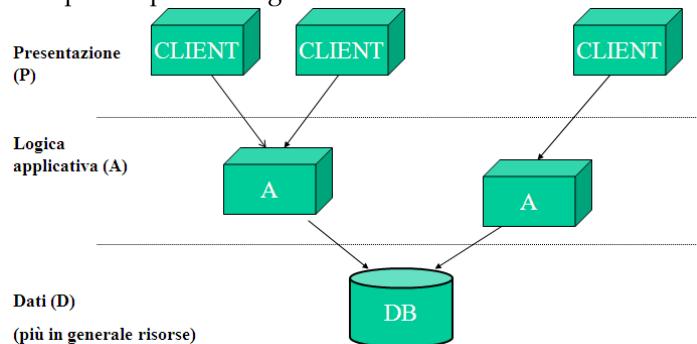
A seguito dell'elevato costo in termini di potenza di calcolo che era richiesto dall'architettura a 2 livelli logici, si è poi sentita l'esigenza di suddividere il livello logico del server in due diversi livelli: la logica applicativa e il dato (o più in generale le risorse necessarie alla logica applicativa). Oltre a tali livelli, continua ad esistere il livello di presentazione (client). Si definiscono perciò i tre livelli seguenti:

- ◆ *Layer di presentazione (P)*
È il livello che si occupa di gestire la logica di presentazione, ovvero le modalità di interazione con l'utente. In altri termini, contiene le modalità di interfacciamento e di rendering delle informazioni. Questo livello è detto anche *front-end* delle applicazioni.
- ◆ *Layer di logica applicativa o logica di business (A)*
Si occupa delle funzioni da mettere a disposizione all'utente.
- ◆ *Layer di accesso ai dati (D)*
Si occupa della gestione dell'informazione, eventualmente con accesso ai database o a sistemi legacy (ovvero sistemi ereditati dal passato, con tecnologie di vecchia generazione).

I livelli 2 e 3 costituiscono, insieme, il *back-end* dell'applicazione.

Questa architettura risponde alla necessità da parte di diverse applicazioni (quindi diversi elementi del livello della logica applicativa) di accedere ad una stessa base di dati (cioè a uno stesso elemento del livello logico dei dati).

L'architettura risulta essere quindi quella in figura:



Questa è l'architettura che oggi viene più frequentemente adottata.

5. Architettura a livelli fisici

Introduzione

Come abbiamo accennato, i livelli fisici sono detti anche *tier*. Un livello fisico è un elaboratore oppure un insieme di elaboratori sul quale vengono allocati dei livelli software. L'utilizzo di diversi livelli fisici consente di aumentare la scalabilità, la flessibilità e la sicurezza del sistema.

Server Farm

Nei moderni sistemi, un singolo tier fisico può essere costituito, anziché da un'unica macchina, da una *server farm*, ovvero da un insieme di elaboratori che condividono il carico elaborativo e le applicazioni e, a seconda delle configurazioni, i dati. I vantaggi derivanti da questa scelta sono:

- ♦ ***La diminuzione dei costi***

I costi sono inferiori, in base al principio del downsizing (costa di più una macchina con certe prestazioni di un insieme di macchine che complessivamente hanno quelle stesse prestazioni).

- ♦ ***L'aumento dell'affidabilità***

Il tier fisico non è accessibile solo se nessuno degli elaboratori che lo costituiscono è accessibile. Di conseguenza, l'affidabilità del tier è data da:

$$a = 1 - \prod_i (1 - a_i)$$

dove a_i è l'affidabilità dell' i -esimo elaboratore della server farm.

- ♦ ***Elevata scalabilità***

È possibile far fronte all'aumento delle richieste semplicemente introducendo nuove macchine nella server farm, a costi non particolarmente elevati (principio del downsizing).

La server farm può essere realizzata in 2 modi diversi:

- ♦ ***Cloning o clonazione***

In questo caso ogni macchina della server farm possiede le stesse applicazioni software e gli stessi dati; le richieste vengono poi distribuite tra i vari cloni sulla base di un sistema di load-balancing. L'insieme di cloni che offrono lo stesso servizio è detto RACS (Reliable Array of Cloned Services).

I RACS possono a loro volta avere 2 diverse configurazioni:

1. ***Configurazione shared-nothing***

In questo caso, i dati memorizzati sono replicati su ogni clone e risiedono in un disco fisso locale al singolo clone. Di conseguenza, un aggiornamento dei dati deve essere applicato ad ogni clone.

2. ***Configurazione shared-disk (cluster)***

In questo caso, esiste un server che gestisce i dischi fissi, e tutti i cloni, quando devono accedere ai dati, devono richiederli al server condiviso.

- ♦ ***Partitioning o partizionamento***

Se la server farm è realizzata per partizionamento, ogni nodo svolge una funzione specializzata. In particolare, le applicazioni e l'hardware vengono replicati sui vari nodi, ma i dati non vengono replicati (ogni nodo possiede una certa porzione dei dati), e quindi ci dovrà essere un sistema di smistamento delle richieste che sia in grado di inoltrare la richiesta al nodo che possiede i dati necessari per soddisfare quella stessa richiesta. Tuttavia, se dovesse guastarsi uno dei nodi della server farm, le funzionalità da esso offerte non sarebbero più disponibili (*graceful degradation*, degrado parziale).

Per evitare il degrado parziale delle funzionalità, anziché usare un singolo nodo per una certa partizione, si può clonare quello stesso nodo. Si ottiene così un RAPS (Reliable Array of Partitioned Service): in questo modo si riescono a preservare sia l'affidabilità che la scalabilità.

Architettura single tiered

L'architettura con un solo livello fisico è oggi usata molto poco. Essa prevede che tutti e 3 i livelli logici (presentazione, applicazione, dati) vengano allocati su un unico mainframe. In parole più semplici, abbiamo su un'unica macchina i dati, le applicazioni e il software di gestione della presentazione.

Questa architettura rappresenta il modello centralizzato e, pur essendo obsoleta ed avendo alcuni limiti come l'assenza di flessibilità, ha ancora una propria validità, perché garantisce livelli di prestazioni, affidabilità e sicurezza che le altre architetture non riescono ad offrire.

Architettura two tiered

- ♦ *Che cos'è*

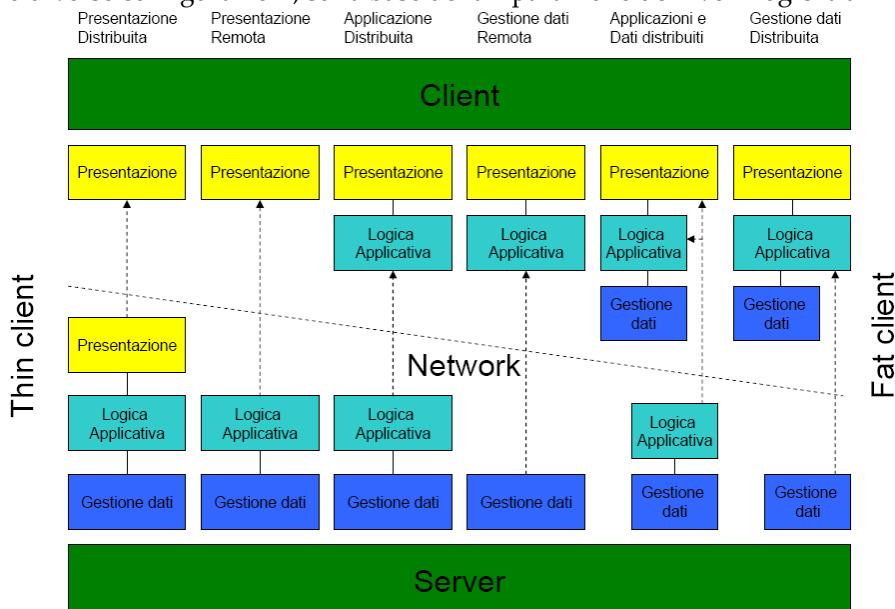
L'architettura two-tiered è un'architettura nella quale si hanno due diversi livelli fisici, tra i quali sono distribuiti i tre livelli logici. I livelli fisici sono rappresentati da:

1. Una stazione di lavoro utente (un PC).
2. Una macchina server.

L'idea-base è di separare da una parte la logica di presentazione, e dall'altra la logica di accesso ai dati. Questa architettura non è particolarmente indicata se la logica applicativa distribuita è molto onerosa dal punto di vista della potenza di calcolo.

- ♦ *Possibili configurazioni*

Sono possibili 6 diverse configurazioni, sulla base della ripartizione dei livelli logici tra i 2 tier:



Le configurazioni, riassunte in figura, differiscono tra loro per alcune caratteristiche come:

- a) Il traffico di rete;
- b) Le prestazioni e i compiti eseguiti dalla stazione di lavoro;
- c) La flessibilità del sistema;
- d) La facilità di modifica.

Un esempio di architettura del tipo *gestione dati distribuita* si ha quando si considera un venditore porta a porta: sul proprio computer il venditore avrà bisogno di avere tutti e 3 i livelli, anche se i dati saranno solo una parte di tutti i dati del sistema informativo, e la restante parte è memorizzata solo sul secondo tier. La necessità di avere dei dati locali è legata al fatto che in alcuni momenti potrebbe non essere disponibile una connessione ad Internet.

La situazione etichettata come *applicazioni e dati distribuiti* è molto simile alla precedente; volendo riprendere l'esempio appena citato, possiamo passare all'architettura con applicazioni e dati distribuiti semplicemente prevedendo che il tier privo del livello di presentazione abbia, ad esempio, un software per il trasferimento dei dati.

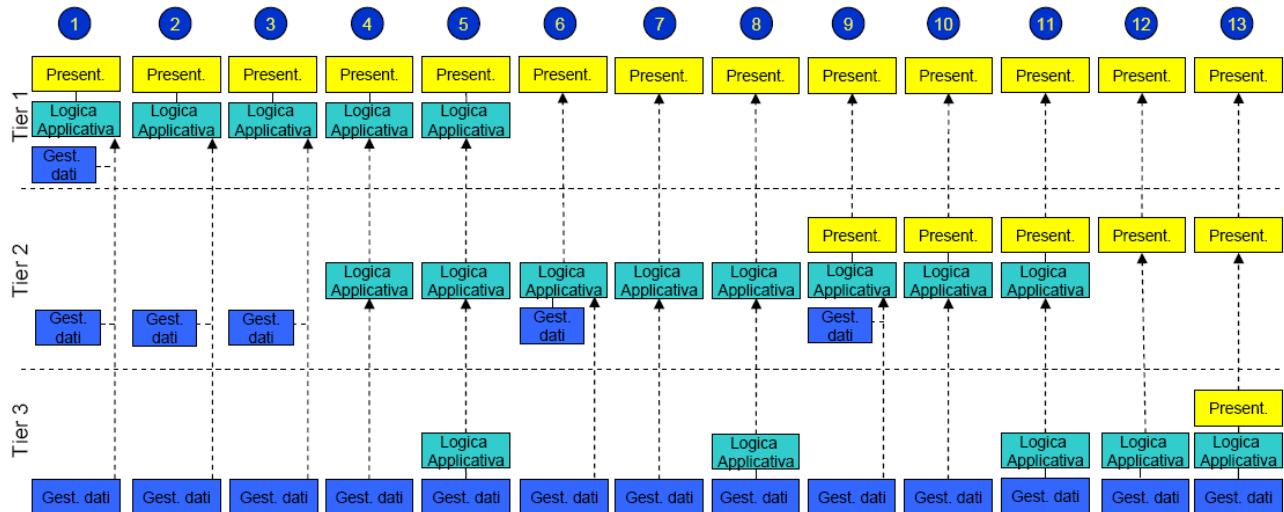
- ♦ *Differenza tra thin client e fat client*

Quando nel client risiede solo il layer di presentazione, si parla di *thin client*; in caso contrario, si parla di *fat client*. Il thin client è quindi un client che ha poca capacità elaborativa, mentre il fat client ha una elevata capacità di elaborazione.

Si osserva che le architetture fat client hanno storicamente comportato svantaggi legati alla necessità di amministrare diverse macchine client anziché un singolo software e al fatto che, in caso di aggiornamento, ogni client ha bisogno di installare l'aggiornamento del software. Questi problemi sono limitati oggi, perché l'utente può installare automaticamente gli aggiornamenti accedendo al web server (sul quale tali aggiornamenti devono essere stati caricati) mediante l'uso del proprio browser. Si devono però imporre alcune limitazioni, relative ad esempio alla versione o al tipo di browser in uso da parte dei client.

Architettura three-tier

In questo caso, i livelli fisici utilizzati sono 3. L'idea di base è quella di dedicare un diverso tier per ogni layer. In realtà però possono esserci anche configurazioni "intermedie", che spesso risultano essere interessanti. Lo schema seguente riassume le possibili configurazioni.



1. Le configurazioni da 1 a 5 sono considerate *fat client* (il client è il tier 1), indicate soprattutto per operare all'interno di una intranet.
2. Le configurazioni da 6 a 13 sono configurazioni *thin client*, adatte ad operare in sistemi aperti come Internet.
3. Tutte le configurazioni nelle quali il layer dati risiede solamente all'interno del tier 3 sono configurazioni particolarmente sicure riguardo all'accesso ai dati da parte degli utenti.

I sistemi a 3 livelli fisici hanno caratteristiche di maggiore scalabilità e flessibilità.

Architettura n-tier

Nelle implementazioni reali si adottano spesso soluzioni con più di 3 tier: gli attuali siti di e-business si sviluppano ad esempio su 5 tier, partizionando ulteriormente su più layer software la logica di presentazione, oppure introducendo dei tier dedicati, a supporto della comunicazione (middleware).

In generale, i livelli fisici aggiuntivi possono essere dedicati ad ospitare sistemi di data warehouse, DBMS, gateway server, e così via. Inoltre, è possibile usare più tier per il livello applicativo, dedicando tier diversi ad applicazioni diverse.

Solitamente, l'introduzione di un maggior numero di tier fisici comporta la riduzione dell'affidabilità complessiva del sistema: l'affidabilità complessiva è infatti data dal prodotto delle affidabilità dei singoli tier e, siccome l'affidabilità è sempre minore di 1, tale produttoria sarà certamente inferiore ai singoli fattori che vi compaiono.

6. Integrazione di sistemi informativi

Introduzione

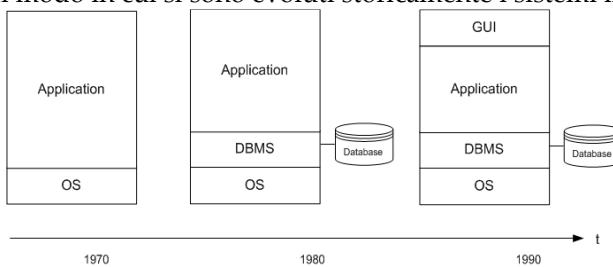
Fino ad ora non abbiamo analizzato le modalità che ci consentono di far colloquiare tra loro diversi sistemi informativi, eventualmente anche caratterizzati da differenti architetture. Secondo le conoscenze che abbiamo al momento, l'unica possibilità è quella di far riferimento alla stessa base di dati, ma chiaramente questo strumento non è sufficiente, perché altrimenti dovrebbe esserci un'unica base di dati integrata per insiemi di organizzazioni anche molto grandi.

Ad esempio, i dati dei cittadini italiani sono dislocati sul territorio, ma non possiamo pensare di creare un'unica base di dati nazionale, anche perché il problema si riproporrebbe poi su scala europea.

Dobbiamo quindi procedere con l'integrazione dei processi.

Evoluzione dei sistemi informativi

La figura seguente mostra il modo in cui si sono evoluti storicamente i sistemi informativi:



Come si osserva, si è passati da sistemi costituiti da un unico livello (*sistemi monolitici*), che si appoggiava direttamente sul sistema operativo, a sistemi con un numero di livelli logici via via superiore.

Diversi livelli di comunicazione tra sistemi

In base anche all'evoluzione storica dei sistemi informativi, possiamo distinguere diverse modalità di interazione tra sistemi diversi:

- ♦ **Comunicazione a livello di presentazione**

Supponiamo ora di voler instaurare un colloquio tra due sistemi monolitici: per le caratteristiche dei sistemi in analisi, non è possibile separare l'interfaccia dalla logica applicativa, perciò l'unica possibilità è quella di creare delle finte interfacce (*wrappers*), che si appoggino sul sistema stesso consentendo l'interazione reciproca mediante le schermate.

Nel caso in cui il sistema non sia monolitico (e quindi sia possibile separare i vari livelli logici), si può comunque creare un wrapper che si appoggia sul livello di presentazione.

- ♦ **Comunicazione a livello di applicazione**

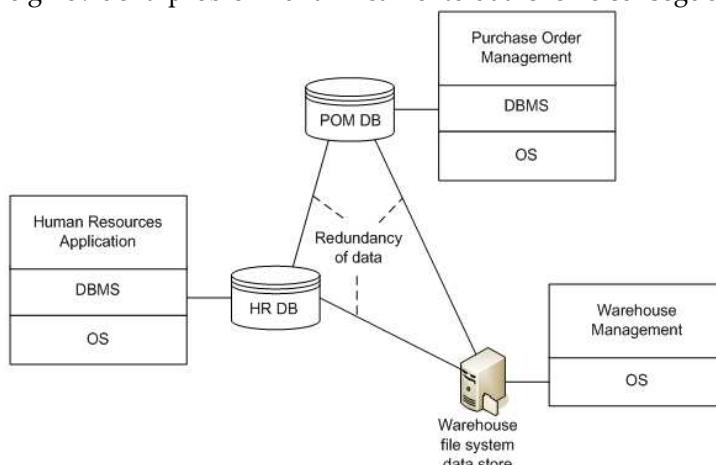
Nel caso in cui sia possibile separare il livello applicativo dall'interfaccia, è possibile anche adottare una soluzione migliore, che consiste nel colloquiare direttamente i livelli applicativi (a patto che il sistema consenta tale comunicazione).

- ♦ **Comunicazione a livello di dati**

Inoltre, se il livello dati è separato dagli altri, è talvolta (ma non sempre) possibile instaurare una comunicazione direttamente al livello dei dati. In generale, possiamo dire che tecnicamente è possibile far passare i dati direttamente da una base di dati ad un'altra, senza passare attraverso i livelli logici superiori, se si verificano le seguenti condizioni:

1. I dati devono essere rappresentati tutti nello stesso modo.
2. Bisogna effettuare dei controlli molto forti.

Nella maggior parte dei casi, queste due condizioni impediscono interazioni al livello dei dati. In genere, possiamo dire che l'interazione al livello dei dati è possibile solo se si tratta di database diversi ma della stessa azienda (questa situazione è rappresentata nella figura sottostante). Tuttavia, abbiamo già detto ampiamente che è preferibile una gestione integrata di tutti i dati dell'azienda, che consenta di evitare la ridondanza e gli evidenti problemi di allineamento dati che ne conseguono.

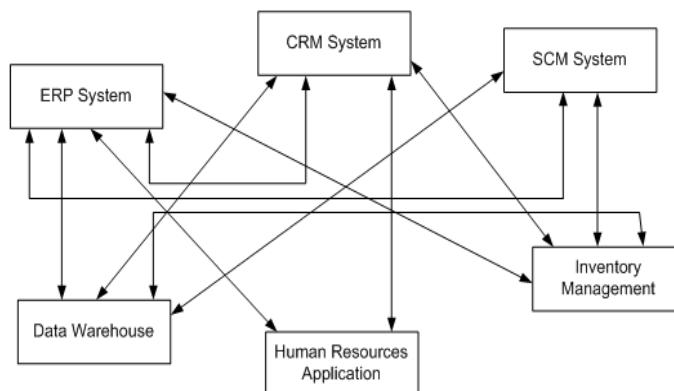


Possibili modalità di comunicazione tra diversi sistemi informativi

Abbiamo appena distinto i vari livelli ai quali la comunicazione può avvenire. Ma, indipendentemente da questo, quali tecniche si possono adottare per far comunicare i diversi sistemi informativi? Vediamo quali sono le diverse possibilità.

- ♦ **Comunicazione diretta tra coppie di sistemi informativi**

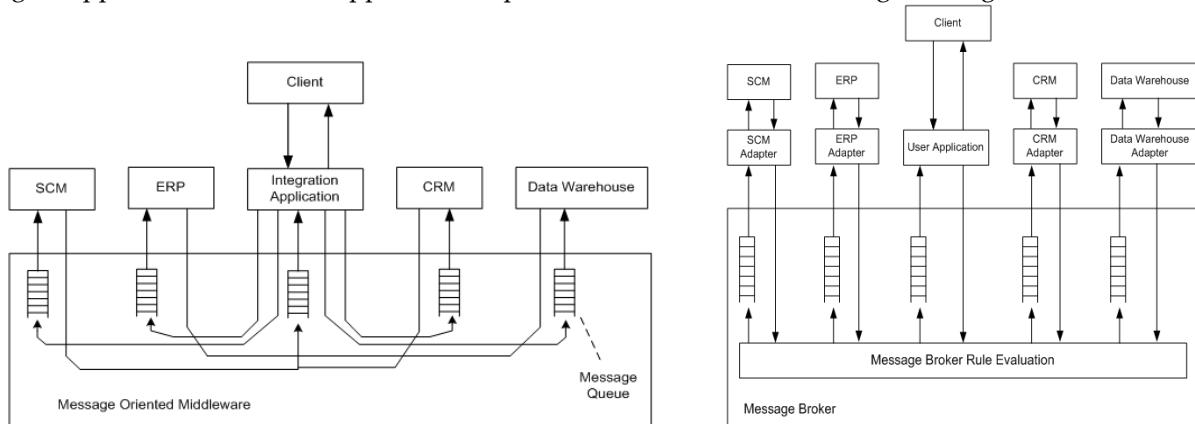
L'integrazione delle applicazioni potrebbe avvenire semplicemente realizzando sistemi separati che comunicano tra loro in maniera diretta. In sostanza quindi si deve definire un opportuno modo di interfacciamento per ogni coppia di sistemi informativi che devono colloquiare tra loro: si definiranno delle regole di comunicazione tra ogni singola coppia di sistemi informativi, e si definiranno i relativi moduli di comunicazione.



Tale scenario, che storicamente è stato il primo ad essere adottato, risulta molto complesso da gestire, anche se consente di definire interfacce specifiche tra i vari sistemi (cosa che in alcune situazioni potrebbe essere utile).

- ♦ **Uso del middleware**

La comunicazione tra i moduli può anche essere resa possibile mediante l'uso di un middleware. Un middleware è un'infrastruttura che consente lo scambio di messaggi tra i vari moduli, secondo delle regole opportune. Possiamo rappresentare questa situazione mediante la seguente figura:



In questo modo, la comunicazione risulta virtualizzata: non si ha una comunicazione diretta tra due moduli, ma si passa attraverso il middleware.

L'uso del middleware presenta come importante vantaggio la possibilità di fornire delle maggiori garanzie, ad esempio nel campo dell'affidabilità: il middleware potrebbe garantire al modulo mittente che il messaggio inviato venga ricevuto dal destinatario (sarà quindi il middleware a gestire l'eventuale ritrasmissione del messaggio).

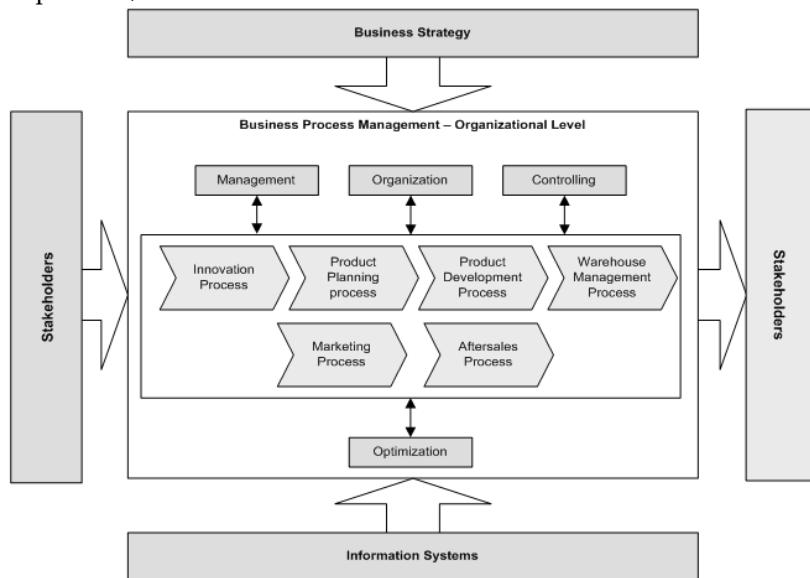
Lo svantaggio che ne deriva è invece rappresentato dalla scarsa flessibilità, e dalla necessità di definire degli adattatori in grado di unificare i formati dei dati e consentire così il colloquio tra i vari sistemi.

Il sistema di middleware può anche essere usato per definire dei sistemi di notifica: quando vengono ricevuti alcuni tipi di messaggi, viene segnalato il corrispondente evento. A svolgere questo compito è un componente del middleware, detto *message broker*.

- ♦ **Integrazione a livello di workflow**

Un altro tipo di integrazione è quella che passa attraverso i sistemi di workflow. Tali sistemi vengono usati sia per la comunicazione interna all'azienda, sia per la comunicazione con l'esterno.

L'integrazione a livello di workflow avviene mediante la codifica delle attività che costituiscono i processi. Questo livello di integrazione avviene quindi al livello di business. In sostanza, l'azienda viene suddivisa in macro-processi, secondo lo schema della catena del valore di Porter:

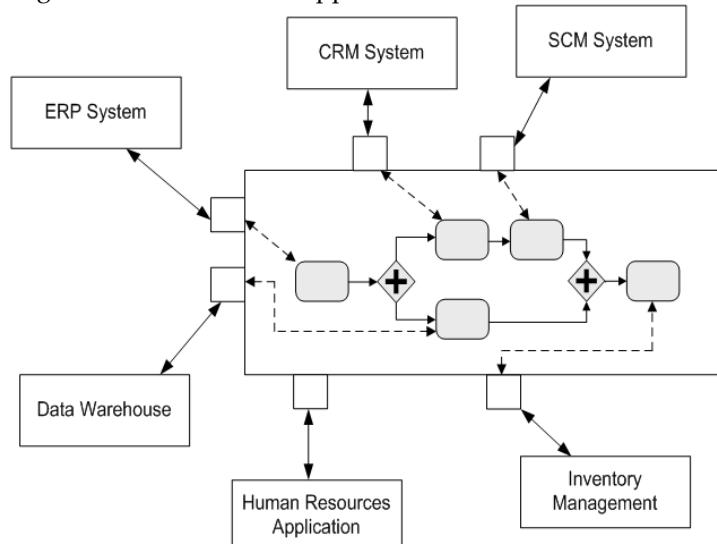


Si definiscono poi i collegamenti tra i vari processi e i collegamenti verso l'esterno. Per fare ciò occorre specificare quali sono i dati che vengono scambiati (ma ad un livello di dettaglio molto diverso rispetto a quello del sistema a messaggi, ovvero con un'astrazione molto superiore), e i passi che devono essere compiuti per mettere in atto questa integrazione.

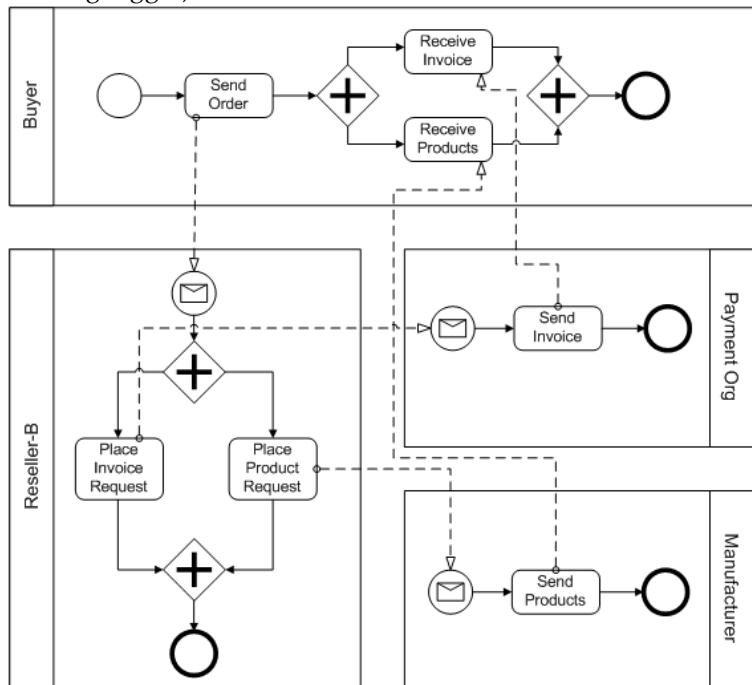
Da un punto di vista tecnologico, l'approccio che possiamo usare è quello della modellizzazione, attraverso il BPMN (Business Process Modeling Notation).

Il livello di astrazione delle attività nello schema BPMN resta generalmente abbastanza alto (anche se poi è possibile scomporre ogni attività in un nuovo schema BPMN che metta in evidenza le attività che a sua volta la costituiscono).

Lo scenario di integrazione mediante modellazione dei processi è rappresentato nella figura seguente, che mette in evidenza come venga definito (mediante BPMN) un processo (workflow) che sia in grado di stabilire quando vengono invocate le varie applicazioni del sistema informativo.



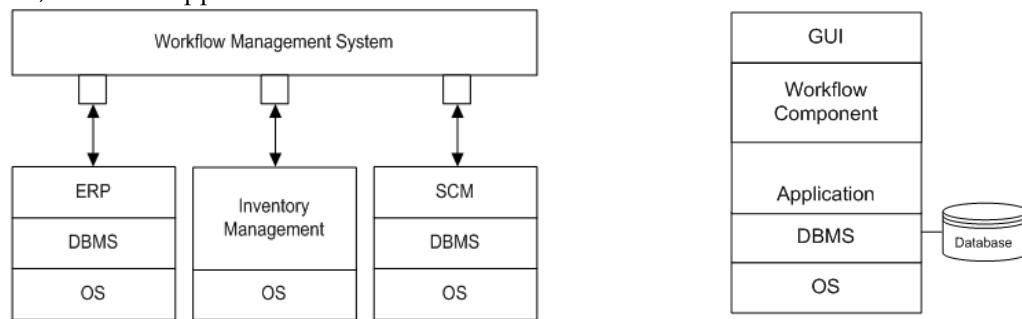
La figura seguente mostra un esempio di BPMN (in seguito verranno descritte nel dettaglio le caratteristiche di questo linguaggio):



Gli schemi BPMN, in sintesi, servono per rappresentare:

1. Le interazioni tra organizzazioni;
2. Gli scambi di messaggi, ma a livello molto astratto;
3. La descrizione dei singoli processi (a volte automatizzabili, altre volte no).

In alcuni casi, i sistemi di workflow si usano solamente per integrare tra loro diversi sistemi informativi della stessa organizzazione (come nella figura a sinistra); altre volte, essi vengono usati anche per l'integrazione tra diversi moduli di una singola applicazione (come mostrato nella figura di destra). In questo caso, si scrive l'applicazione mediante l'uso di un motore di workflow.



Ovviamente, anche all'interno dei sistemi middleware possono esserci delle regole nello stile dell'ottica di processo, perché le tecnologie non sono completamente separate.

Si noti però che BPMN ha alcune importanti limitazioni. Ad esempio, non include i modelli dei dati, le regole di business, i meccanismi di archiviazione, la struttura dell'organizzazione e le risorse a disposizione, ...

♦ *Uso di XML*

Oltre ad usare dei wrapper per consentire la comunicazione tra sistemi, qualora non sia possibile separare bene l'interfaccia dall'applicazione, è possibile passare all'ottica a servizi (ad esempio, servizi web). I dati vengono quindi rappresentati mediante XML per lo scambio tra le varie applicazioni. Passare all'ottica a servizi significa cambiare la sequenza di interazione e l'ottica dell'interfaccia, che è pensata per applicazioni e non più per utenti.

Dobbiamo però aggiungere ai sistemi precedenti un nuovo livello al posto della presentazione.

Capitolo 5: I Workflow e WfMS

1. Che cosa si intende per workflow?

Il concetto di workflow

Un workflow è una sequenza di operazioni interconnesse tra loro, dichiarate come compiti assegnati a persone singole oppure a gruppi di persone all'interno di una stessa organizzazione, finalizzate all'esecuzione di un certo processo.

Un workflow è quindi l'automazione (totale o parziale) di un processo di business nel quale i documenti, le informazioni o i compiti sono passati da un partecipante ad un altro per svolgere attività, secondo un insieme di regole procedurali.

Molte singole istanze di processo possono essere operative durante l'enactment (messa in atto) di un processo (o "caso" o "workflow case").

Utilizzare la tecnologia dei workflow significa quindi automatizzare i processi che coinvolgono una combinazione di attività umane ed automatiche, in particolare con il coinvolgimento di applicazioni e strumenti IT.

Come definire un processo

Dal concetto di workflow, risulta chiaro che la tecnologia del workflow richiede la definizione rigorosa dei processi di business. Vediamo allora come si può definire un processo. Per farlo, occorre individuare:

1. La rete di attività che lo costituiscono.
2. Quali sono i criteri per iniziare e per terminare il processo stesso.
3. Quali sono le informazioni sulle singole attività.
4. Chi sono i partecipanti al processo, dove il termine *partecipante* si può riferire sia ad una *risorsa umana* (cioè una persona fisica o un gruppo di persone), sia ad un'applicazione software.
5. Quali sono i documenti e i dati legati alle attività del processo. In genere però non vengono rappresentati nello schema dei flussi di dati.

Una volta individuati tutti questi elementi, bisogna definire il processo anche in una maniera formale, rappresentandolo in una forma che ne consenta la manipolazione automatica, per la modellazione e per la sua messa in atto. I costrutti di base che si utilizzano per la definizione della rete di attività sono:

- a) La sequenza (cioè due attività vengono eseguite l'una dopo l'altra).
- b) Il parallelo (due attività in parallelo sono 2 attività che vengono eseguite contemporaneamente).
- c) L'alternativa (due attività sono in alternativa se ne viene eseguita solo una, sulla base di opportune condizioni, che possono essere di vario tipo).

Possiamo quindi dire che il processo di business è "ciò che si vuole che accada", mentre la definizione del processo di business è la rappresentazione di ciò che si vuole che accada. All'interno di questa rappresentazione, potremo avere le definizioni di sottoprocessi. Inoltre, ogni processo è costituito da un insieme di attività, che possono essere manuali (e quindi non gestite da alcun sistema automatizzato) oppure automatizzate.

Istanza di processo (o casi, o workflow case)

In maniera informale, un'*istanza di processo* è una rappresentazione di ciò che sta realmente accadendo, e rispetta quanto è stato dettato dalle definizioni del processo al quale corrisponde. Ogni volta che viene avviata l'esecuzione di un processo, si avvia quindi una nuova istanza di quel processo. In generale, è possibile che siano attive allo stesso tempo diverse istanze di processo, operanti su dati e/o caratterizzate da partecipanti diversi.

Un'istanza di processo comprende una o più istanze di attività, ciascuna delle quali può comprendere uno o più:

- a) *Elementi di lavoro (work items)*

Un elemento di lavoro è un task allocato ad uno specifico partecipante.

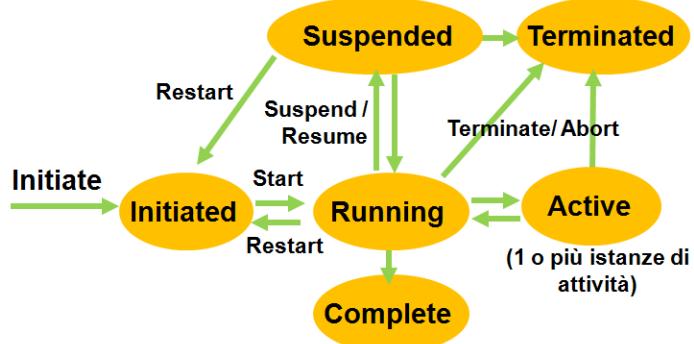
- b) *Applicazioni invocate*

Le applicazioni invocate sono gli strumenti e le applicazioni informatiche di supporto.

2. Stati di processi e attività

Gli stati di un'istanza di processo

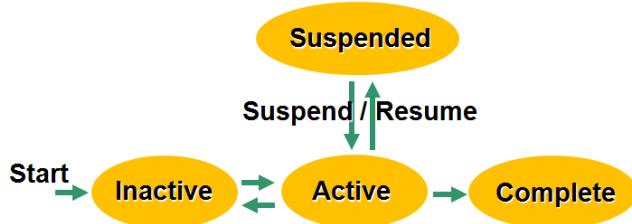
In maniera molto simile a quanto accade per i processi informatici, i processi di business possono trovarsi in diversi stati. Il seguente diagramma mette in evidenza le transizioni da uno stato all'altro.



Inizialmente, il processo si trova in stato "initiated", cioè è pronto ad essere eseguito, ma la sua esecuzione non è ancora stata avviata. Dopo l'operazione di start, si passa allo stato "running": il processo è quindi in esecuzione. Quando poi si hanno in esecuzione una o più istanze di attività, il processo passa allo stato "active". Dallo stato active, si può tornare allo stato running, quando termina l'attività in esecuzione (o terminano le attività in esecuzione) e ancora non è stata avviata la successiva. Il processo può poi essere terminato (tipicamente per un errore), e in tal caso si passa dallo stato active o running allo stato terminated. Quando invece si è tornati allo stato running perché è terminata l'ultima attività del processo, allora si passerà allo stato "complete"(il processo cioè è terminato con successo). Dallo stato running è anche possibile passare allo stato "suspended": il processo viene cioè sospeso per un certo intervallo di tempo, al termine del quale viene riavviato da capo (tramite restart), oppure si ritorna allo stato running. Infine, dallo stato running si può tornare allo stato initiated se si riavvia dall'inizio il processo in esecuzione.

Transizioni di stato di un'attività

Il seguente diagramma inoltre rappresenta in maniera intuitiva quali sono le transizioni di stato che si possono avere per una singola attività di un processo:



L'attività è in stato "active" se ha almeno una *istanza di lavoro*, ovvero qualcuno la sta realmente eseguendo.

3. Caratteristiche di qualità dei processi

Vediamo adesso quali sono le caratteristiche fondamentali per la valutazione della qualità di un processo.

Economicità

Una delle caratteristiche fondamentali per la qualità di un processo è l'economicità: un buon processo deve infatti essere in grado di fornire un servizio al più basso costo possibile, in relazione ad una specifica qualità desiderata per l'output o per il risultato finale.

Efficienza

L'efficienza di un processo è data dal rapporto tra gli input e gli output. In altri termini, è data dal rapporto tra le risorse e i costi da un lato, e i carichi di lavoro dall'altro.

Un processo quindi è efficiente se consente di raggiungere il risultato prefissato con uno sforzo ridotto (bassi costi e risorse richieste). L'efficienza rappresenta una caratteristica che viene ben espressa dalla frase "doing the things right": un processo è efficiente se consente di fare le cose "bene", secondo l'accezione che abbiamo appena dato di questo termine.

Efficacia

L'efficacia di un processo è data invece dal rapporto tra il risultato finale e gli obiettivi iniziali. In altri termini, l'efficacia è una misura della capacità dell'azienda di soddisfare le richieste che provengono dai clienti.

In contrasto con quanto detto a riguardo dell'efficienza, possiamo esprimere il concetto di efficacia mediante l'espressione "doing the right thing": un processo è efficace se fa "la cosa giusta", ovvero quella che desidera l'utente.

Osservazione

È bene fare attenzione alla differenza tra efficacia ed efficienza, perché si tratta di concetti molto diversi tra loro, ma facili da confondere. Ad esempio, è possibile che un'azienda (o un processo) abbia elevata efficienza, mentre la sua efficacia è molto bassa.

4. I Workflow Management Systems (WfMS)

Gestione dei flussi di lavoro

In processi complessi, la pianificazione e la gestione delle diverse attività diventa sempre più difficile, al punto che controllare i flussi di lavoro può richiedere tanto tempo quanto l'esecuzione del lavoro stesso.

La soluzione a questo problema sta nell'utilizzo di strumenti software, detti WfMS (Workflow Management System). Un WfMS gestisce attivamente il coordinamento di attività tra persone in processi di business generici.

Cos'è un workflow management system?

I Workflow Management System (WfMS), come dice il nome stesso, sono dei sistemi software per la gestione di workflow (flussi di lavoro). In particolare, i WfMS interpretano delle regole procedurali, sulla base delle quali gestiscono lo scambio di informazioni, lavoro o documenti all'interno dell'azienda e, quando è possibile, automatizzando le attività da svolgere. Possiamo quindi dire che il WfMS è il sistema software che controlla gli aspetti automatizzati di un processo di business.

Un WfMS può gestire nello stesso istante diverse istanze di uno stesso processo (ovvero lo stesso processo viene eseguito più volte contemporaneamente, ma con dei dati e/o dei partecipanti diversi).

Funzioni di un WfMS

Le funzioni interne di un WfMS possono essere raggruppate in diverse aree funzionali:

- ◆ ***Build-time functions***

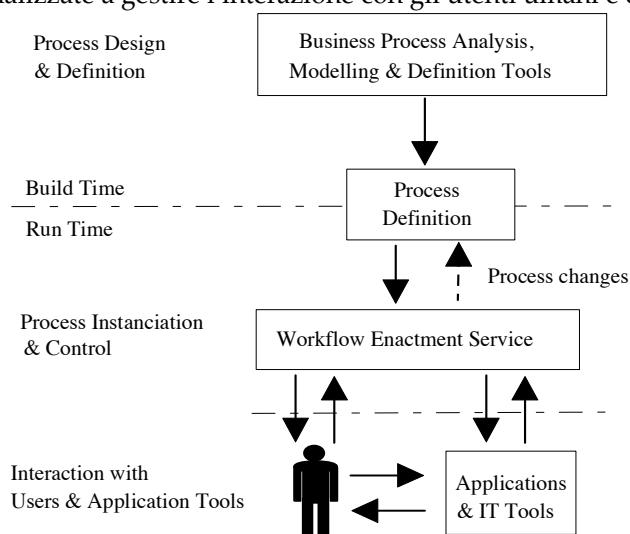
Sono le funzioni che riguardano la definizione e la modellazione dei processi di workflow e delle attività che li costituiscono.

- ◆ ***Run-time control functions***

Sono le funzioni che riguardano la gestione dei processi di workflow in un ambiente operativo e la sequenzializzazione delle attività gestite come parte del processo.

- ◆ ***Run-time interactions with human users and IT application tools***

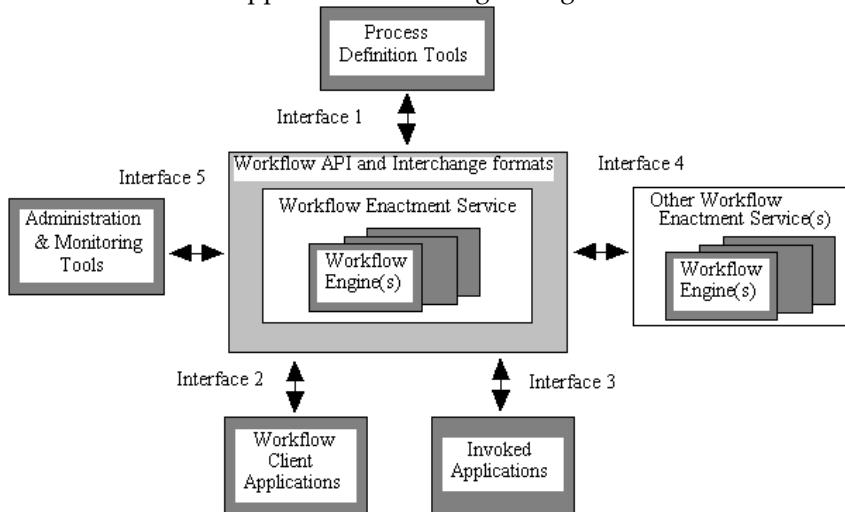
Si tratta di funzioni finalizzate a gestire l'interazione con gli utenti umani e con altri strumenti software.



5. Modello di riferimento per la struttura di un WfMS

Introduzione

Vogliamo ora introdurre il *workflow reference model*, ovvero il modello di riferimento per l'architettura dei sistemi di workflow. Tale modello è rappresentato nella figura seguente:



Come mostrato dalla figura, il WfMS possiede al suo interno un certo numero di componenti che interagiscono tra loro in maniera predefinita. Al fine di consentire l'interoperabilità tra tali componenti, sono state definite delle interfacce standardizzate per lo scambio di dati tra i componenti stessi.

Workflow Enactment Service (WES)

- **Cos'è un WES?**

Il *Workflow Enactment Service* è quella parte del WfMS che fornisce l'ambiente per l'esecuzione delle singole istanze dei processi. Possiamo quindi dire che il WES è un servizio software finalizzato alla creazione, gestione ed esecuzione di istanze di processo. Un WES è costituito da uno o più *workflow engine*.

- **Workflow engine**

Un *workflow engine* è un'applicazione software che si occupa di interpretare, gestire ed eseguire una parte di un processo di business (o, al limite, l'intero processo di business). In particolare, l'engine determina quali sono le attività da mandare in esecuzione.

Per fare ciò, esso deve interpretare gli eventi che si verificano (ad esempio, l'invio di documenti ad un server), e compiere delle azioni di risposta a tali eventi, in accordo ai processi definiti. Le azioni possono essere di ogni tipo (ad esempio, il salvataggio del documento in un apposito sistema, l'assegnazione di nuove attività mediante l'invio di e-mail agli utenti, ...).

Tipicamente, un workflow engine fornisce i servizi per:

1. Interpretare la definizione del processo;
2. Controllare le istanze di processo (ovvero gestire la creazione, l'attivazione, la sospensione, la terminazione, ... cioè i vari passaggi da uno stato ad un altro).
3. Navigare tra le varie attività di processo, che potrebbero coinvolgere operazioni da svolgersi in maniera sequenziale o in parallelo.
4. Consentire l'ingresso e l'uscita di specifici partecipanti.
5. Identificare le attività da sottoporre all'attenzione degli utenti e mettere a disposizione un'interfaccia per supportare l'interazione con gli utenti.
6. Mantenere i dati di controllo e i dati salienti relativi al workflow, mediante lo scambio di tali dati con applicazioni o utenti.
7. Mettere a disposizione un'interfaccia per invocare applicazioni esterne e collegare tutti i dati rilevanti sul workflow.
8. Eseguire azioni di supervisione, per fini di controllo, amministrazione e verifica.

- ♦ **Osservazione**

Nonostante il workflow enactment service sia stato descritto come un'entità logica singola, in realtà esso può essere un sistema centralizzato, oppure un sistema distribuito. In questo secondo caso, diversi workflow engine controlleranno ciascuno una diversa parte della messa in atto del processo e interagiranno con quel sottoinsieme di utenti e applicazioni che sono legati alle attività di quella "fetta di processo" di cui sono responsabili.

I sistemi di workflow distribuiti fanno uso di specifici protocolli e formati di scambio dati tra i workflow engine, al fine di sincronizzare le operazioni e scambiare informazioni di controllo del processo e delle attività.

Altri componenti

Gli altri componenti del WfMS sono:

- ♦ **L'interfaccia WAPI**

Le applicazioni possono interagire con i servizi offerti dal Workflow Enactment Service per mezzo di un'interfaccia detta Workflow API (o WAPI). Essa mette quindi a disposizione un modo per invocare i servizi del WES.

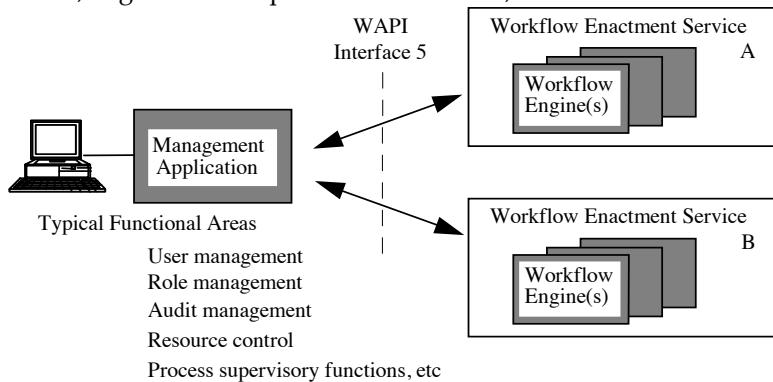
- ♦ **Process definition tools**

Contiene:

1. Il *process model designer*, che è uno strumento software per la definizione dei modelli dei processi.
2. Il *process model repository*, che è l'archivio dei modelli dei processi registrati nel WfMS.

- ♦ **Administration & monitoring tools**

Si tratta di un'applicazione usata per amministrare il WfMS e per monitorarne il funzionamento. Alcune sue tipiche aree funzionali sono la gestione degli utenti e dei ruoli, il controllo delle risorse, la supervisione dei processi, la gestione di operazioni di verifica, ...



- ♦ **Workflow client application (worklist handler)**

Come abbiamo detto, il workflow engine determina quali sono le attività da mandare in esecuzione e individua quali sono i partecipanti al workflow a doverle svolgere (si occupa di allocare le attività). Esso genera così delle worklist (una worklist è l'elenco di work items, ovvero di compiti, assegnati ad un certo partecipante al workflow).

I vari partecipanti (ovvero utenti singoli o gruppi di utenti) devono perciò poter interagire con il workflow engine, al fine di poter ricevere la propria worklist e di mandare all'engine eventuali messaggi di notifica (ad esempio, per segnalare il completamento delle attività ad essi assegnate).

L'applicazione che si occupa di svolgere tali operazioni prende il nome di *worklist handler* o *workflow client application*. Il workflow handler è perciò il sistema software responsabile dell'organizzazione del lavoro per conto di un utente. È il worklist handler ad avere la responsabilità di selezionare e far procedere le singole attività dalla worklist.

- ♦ **La client application interface**

L'interazione tra il workflow engine e il worklist handler avviene mediante un'opportuna interfaccia, detta *client application interface*.

- ♦ **La invoked application interface**

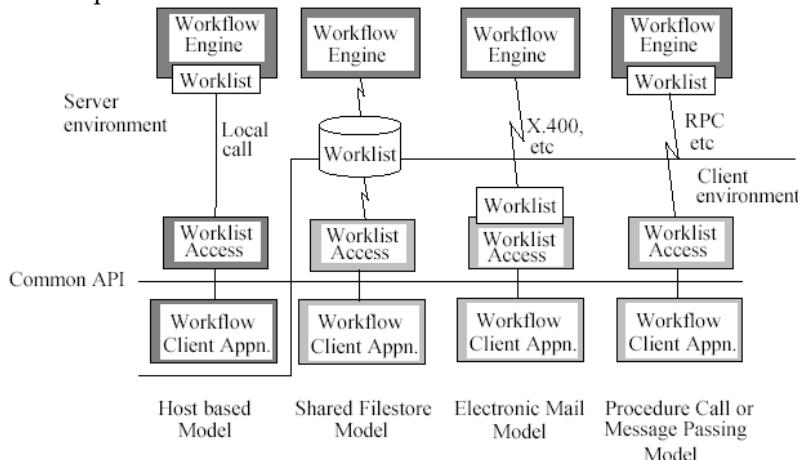
È l'interfaccia che permette al workflow engine di attivare direttamente una specifica applicazione per intraprendere una particolare attività (tipicamente si tratta di un'applicazione server, priva di interfaccia con l'utente).

Diverse implementazioni del worklist handler

Nel modello a workflow, l'interazione tra il worklist handler ed un particolare workflow engine avviene mediante un'interfaccia ben definita. Nel caso più semplice, la worklist è accessibile al workflow engine affinché quest'ultimo possa assegnare delle attività ai vari utenti, ed è accessibile al worklist handler allo scopo di presentare agli utenti le attività ad essi assegnate, in modo che possano poi eseguirle.

Questa interazione può però essere implementata in maniera diversa, a seconda di vari fattori, tra i quali anche il tipo di infrastruttura usata per supportare la distribuzione delle worklist.

La figura seguente mostra 4 possibili scenari:



1. Host based model

In questo modello, la comunicazione tra il client worklist handler e il workflow engine avviene mediante un'interfaccia che si trova su un terminale o su una workstation remota. L'utente deve quindi inviare a tale interfaccia la richiesta di accesso ai dati. L'interfaccia accede poi alla worklist mediante una chiamata locale.

2. Shared filestore model

Il modello *shared filestore model* prevede che si abbia un archivio condiviso, che si trova sul confine tra la piattaforma client e la piattaforma server ed è accessibile ad entrambe. Il worklist handler è quindi implementato come una funzione client che comunica con tale archivio di dati.

3. Electronic Mail Model

La comunicazione avviene per mezzo della posta elettronica, che supporta la distribuzione dei work items (ovvero dei compiti) a singoli partecipante del processo.

4. Procedure call or message passing model

La comunicazione avviene mediante un meccanismo di scambio di messaggi (ad esempio, RPC: Remote Procedure Call). In questo scenario, la worklist potrebbe essere fisicamente allocata nel workflow engine o nel worklist handler, a seconda delle specifiche caratteristiche di implementazione.

6. XPDL

Che cos'è XPDL?

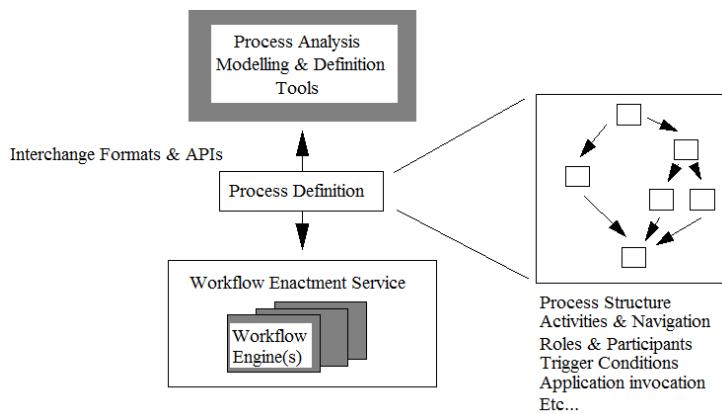
XPDL è l'acronimo per XML Process Definition Language. Come ci suggerisce il nome, si tratta di un linguaggio XML che consente di modellizzare i processi di business.

In particolare, si tratta di un linguaggio standardizzato dalla WfMC, che definisce uno schema standard per la rappresentazione di un processo di business attraverso un *workflow*.

A che cosa serve XPDL?

L'obiettivo del linguaggio XPDL è quello di fornire un punto di riferimento unico per la rappresentazione dei processi di lavoro e per garantire l'interoperabilità tra le applicazioni software che gestiscono tali processi.

Ad esempio, parlando dei WfMS abbiamo introdotto uno dei suoi componenti, che abbiamo indicato come *process definition tools*, il quale comunica (mediante un'opportuna interfaccia) al WES le definizioni dei processi. Senza un linguaggio come XPDL, sarebbe del tutto impossibile rappresentare e comunicare le definizioni dei processi.



La standardizzazione di XPDL

La WfMC diede vita ad un primo linguaggio standard per la definizione dei processi di business nel 1998: tale linguaggio era chiamato WPDL (Workflow Process Definition Language).

Poco dopo però nacque XML, così la WfMC si rese conto che sarebbe stato utile che il proprio linguaggio standard fosse un linguaggio XML: così nel 2002 diede vita ad una seconda versione del proprio linguaggio per la definizione dei processi, chiamata XPDL 1.0. Nel 2005 è stata rilasciata la versione XPDL 2.0 e nel 2008 la versione XPDL 2.1.

I documenti di standardizzazione di XPDL includono un meta-modello per la descrizione delle definizioni di processo, ma anche un DTD per lo scambio delle definizioni di processo. Il pubblico al quale tali documenti si rivolgono è costituito soprattutto da organizzazioni di produttore che cercano di implementare il linguaggio XPDL.

Relazioni tra XPDL e BPMN

Come abbiamo detto, un altro modo che abbiamo a disposizione per descrivere una definizione di processo è quello di usare il linguaggio BPMN, che però è un linguaggio grafico e quindi non è adatto allo scambio di informazioni tra applicazioni.

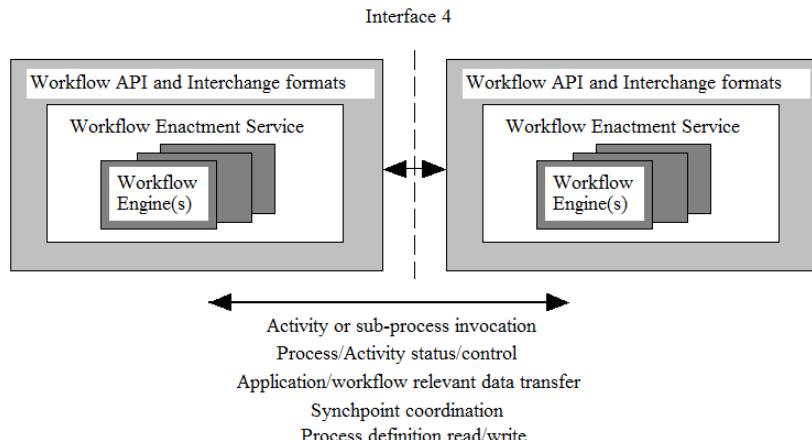
Di fatto, XPDL ci consente di serializzare la notazione grafica BPMN. In particolare, a partire dalla versione XPDL 2.0, tutti i formalismi di BPMN possono essere rappresentati anche con XPDL (ciò non era vero per le precedenti versioni: il linguaggio XPDL 1.0 aveva una capacità espressiva inferiore di BPMN).

7. Interoperabilità tra diversi sistemi di workflow

Introduzione

Uno degli obiettivi della WfMC è quello di definire degli standard per consentire a sistemi di workflow realizzati da diversi produttori software di scambiarsi reciprocamente degli elementi di lavoro.

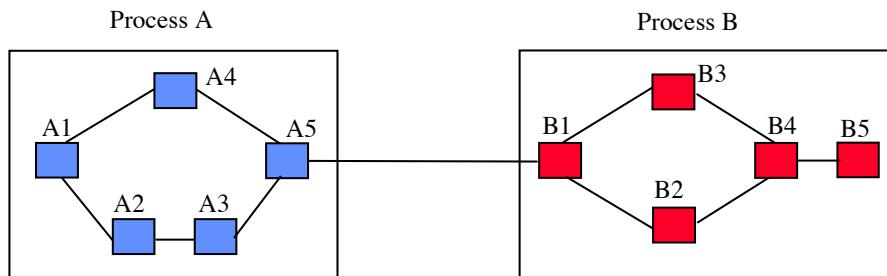
Parlando della struttura di un WfMS abbiamo infatti messo in evidenza la presenza di un'interfaccia (l'interfaccia 4) finalizzata allo scambio di informazioni tra diversi sistemi di workflow.



Per consentire l'interoperabilità, sono stati identificati 4 diversi possibili modelli di interoperabilità, con possibilità diverse.

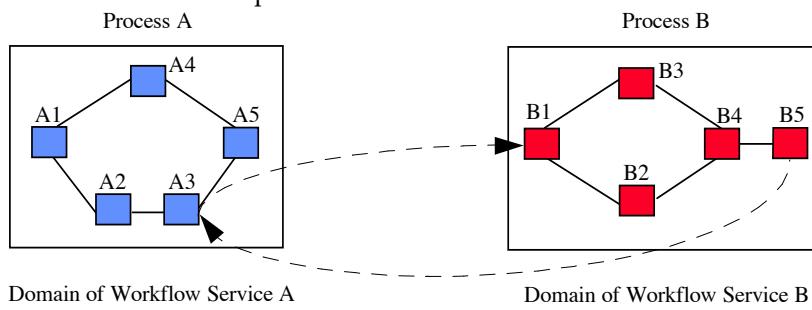
Il modello a servizi concatenati

Questo modello consente il trasferimento di un singolo elemento di lavoro (istanza di processo o attività) tra due ambienti WFMS diversi, che operano indipendentemente dopo lo scambio, senza ulteriori sincronizzazioni.



Il modello a sottoprocessi annidati

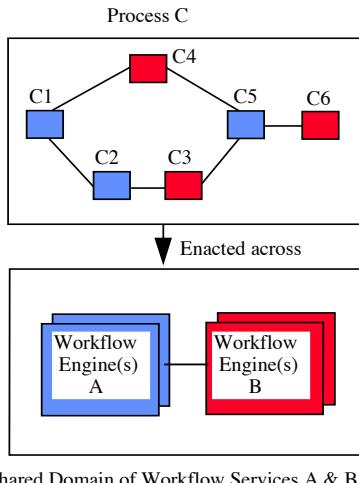
In questo modello, un sistema di workflow ha delle attività che vengono eseguite come processi completi sull'altro sistema di workflow. Ad esempio:



Nel diagramma, il servizio di workflow A ha un'attività definita, A3, che viene eseguita come un processo completo (B) sul sistema di workflow B, con ritorno del controllo ad A dopo l'esecuzione.

Modello peer-to-peer

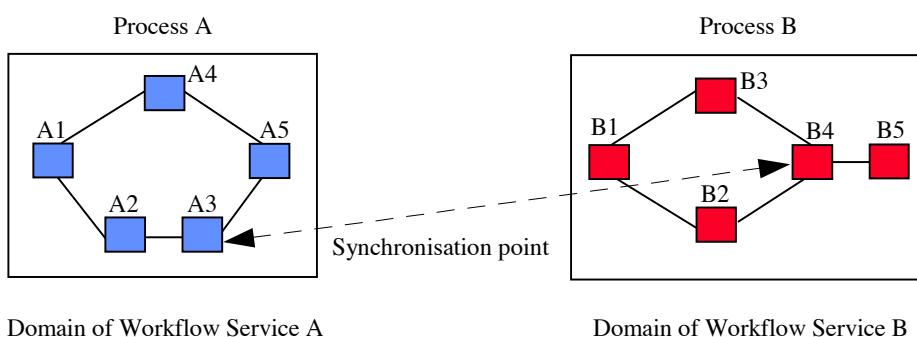
Questa alternativa richiede che entrambi i servizi di workflow supportino un'interfaccia API comune per la comunicazione e che siano in grado di interpretare una definizione di processo comune, che può essere importata da un processo di definizione esterno o trasferita a run-time durante l'esecuzione.



Alcune attività quindi vengono eseguite da un sistema di workflow, altre vengono eseguite dall'altro.

Modello parallelo sincronizzato

I due processi operano sostanzialmente in maniera indipendente, ma sono richiesti dei punti di sincronizzazione tra i due processi stessi.



Capitolo 6: Ripasso di UML

1. UML

Prima di studiare il linguaggio BPMN, al quale abbiamo già fatto riferimento più volte, ripassiamo il linguaggio UML, che è stato introdotto già in altri corsi di studio precedentemente affrontati.

Che cos'è UML

UML (acronimo di Unified Modeling Language) è un linguaggio semiformale di specifica, che fornisce dei modelli grafici utili ogni fase della progettazione di sistemi software.

Tra i tanti diagrammi che sono compresi in UML, sono di interesse ai fini del nostro corso:

1. I class diagram;
2. I sequence diagram;
3. Gli use case diagram;
4. Gli activity diagram.

2. Diagrammi dei casi d'uso

Cosa sono?

I diagrammi dei casi d'uso (detti anche *use case diagram*) sono dei diagrammi che definiscono:

1. Le funzionalità di un sistema informativo (cioè i suoi processi);
2. Come il sistema informativo agisce e reagisce.

In particolare, i diagrammi dei casi d'uso descrivono:

1. Il sistema;
2. L'ambiente;
3. Le relazioni tra il sistema e l'ambiente.

In altre parole, posiamo dire che il diagramma dei casi d'uso ci permette di definire ciò che c'è all'interno di un sistema informativo e ciò che è al di fuori del sistema stesso, di indicare le funzionalità del sistema informativo, quali sono i personaggi che sanno usare tali funzionalità e quali sono gli elementi di cui occorre disporre per poter usare queste funzionalità, oltre che le relazioni tra gli attori (il termine attore è usato per indicare chi usa le funzionalità del sistema informativo).

Il sistema può essere definito a diversi livelli di granularità. L'obiettivo dello use case diagram è identificare le funzionalità fornite da un sistema. Sono indicate solo le funzionalità visibili.

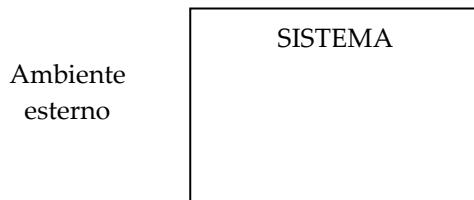
Gli elementi dello use case diagram

Sulla base di quanto abbiamo appena detto, uno use case diagram è costituito dai seguenti elementi:

1. Il sistema;
2. Gli use case;
3. Gli attori;
4. Le relazioni tra gli attori e gli use case;
5. Le relazioni tra use case e use case.

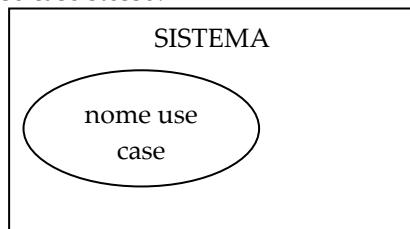
Il sistema e gli use case

Gli use case elencano le funzionalità a disposizione, che vengono fornite dal "sistema". In uno use case diagram il sistema è identificato da un rettangolo che racchiude gli use case relativi alle funzionalità fornite. Il rettangolo è accompagnato da una label, all'interno della quale è indicato il nome del sistema stesso.



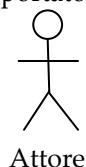
Il rettangolo che rappresenta il sistema quindi separa ciò che è all'interno del sistema stesso da ciò che invece appartiene al mondo esterno.

All'interno del sistema vengono indicati gli use case, ciascuno dei quali rappresenta una particolare funzionalità messa a disposizione dal sistema stesso. Il punto di vista con il quale devono essere definiti gli use case è sempre quello dell'utilizzatore. Ogni use case è rappresentato mediante un ovale all'interno del quale viene indicato il nome dello use case stesso.



Attori

Come accennato, gli attori sono gli utilizzatori del sistema. Ogni attore è rappresentato mediante il seguente simbolo, con un'etichetta nella quale è riportato il nome (o una descrizione) dell'attore stesso:



Attore

Più nel dettaglio, possiamo dire che un attore è un'entità esterna (una persona, ma potrebbe anche essere un sistema hardware esterno) che interagisce con il sistema, assumendo un certo ruolo e che:

1. Controlla le funzionalità del sistema informativo;
2. Fornisce input o riceve output dal sistema.

Da quanto detto, consegue che:

- a) Possono esserci diversi attori con lo stesso ruolo, ma anche diversi ruoli con lo stesso attore;
- b) Diversi attori possono esercitare su uno stesso use case.

Gli attori possono essere il mezzo per individuare gli use case (si individua ciò una lista di attori).

Nell'interazione tra attore e sistema, l'attore può rivestire un ruolo attivo (cioè richiede delle funzionalità oppure passivo (cioè è il sistema che, per svolgere una sua funzione, ha bisogno dell'attore).

Relazioni tra gli elementi di uno use case diagram

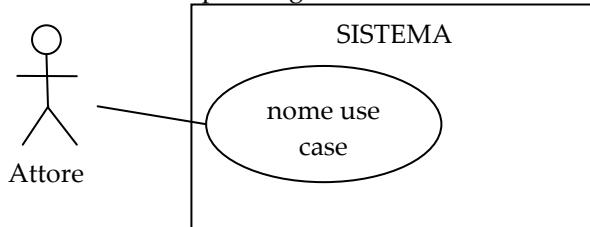
Abbiamo introdotto già tutti gli elementi che possono comparire in uno use case diagram. Tuttavia, la parte forse più significativa di uno use case diagram è rappresentata dai legami che esistono tra tali elementi, e che possono essere di tre tipo:

1. Legami tra diversi use case dello stesso sistema;
2. Legami tra uno use case ed un attore;
3. Legami tra attori.

Legami tra attori e use case: associazione

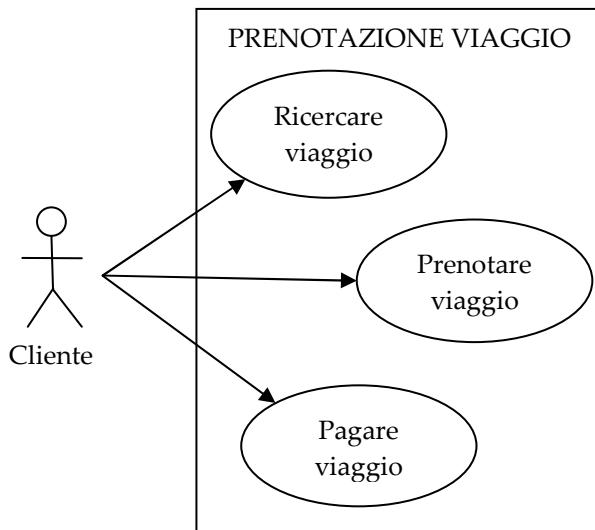
Iniziamo analizzando il modo in cui possono essere legati tra loro un attore ed uno use case. Tale relazione, detta *associazione*, è una relazione di "uso": l'attore nei confronti dello use case può essere il beneficiario, il controllore, può essere informato,

L'associazione si rappresenta mediante un semplice segmento:



L'associazione può anche avere un verso, che è rappresentato orientando il segmento mediante una freccia. In particolare, la freccia va verso lo use case se l'attore è il soggetto attivo dell'associazione (cioè è l'attore a richiedere la funzionalità), mentre ha verso opposto se l'attore è soggetto passivo (cioè è lo use case che richiede l'intervento dell'attore).

Ad esempio:

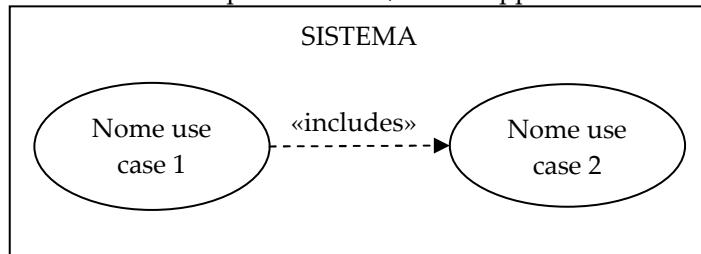


Legami tra diversi use case: associazione

Nel precedente esempio, abbiamo definito un sistema con 3 funzionalità completamente scollegate tra loro, senza che sia definito alcun ordine tra di esse (si noti che negli use case non interviene mai la variabile tempo). Tuttavia, tale situazione non modella efficacemente la realtà. Possiamo allora introdurre delle relazioni tra gli use case, le quali mettano in evidenza dei legami tra di essi. Questi tipi di legami possono essere di vario tipo:

- ◆ **Inclusione**

L'inclusione, rappresentata dallo stereotipo «includes», che si rappresenta come:

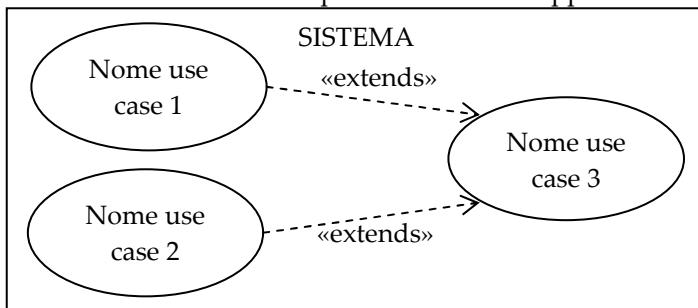


In questo tipo di legame, lo use case dal quale ha origine la freccia ha bisogno, per poter essere completato, dell'esecuzione della funzionalità descritta dallo use case al quale arriva la freccia.

La relazione di inclusione è stata introdotta perché spesso esistono use case che rappresentano delle attività ricorrenti, condivide da use case più complessi; per evitare di ripetere in ogni use case la descrizione dell'attività comune, la si mette a fattor comun, indicando che viene inclusa in uno o più use case più complessi (il meccanismo è quindi analogo a quello della chiamata ad una sottoprocedura).

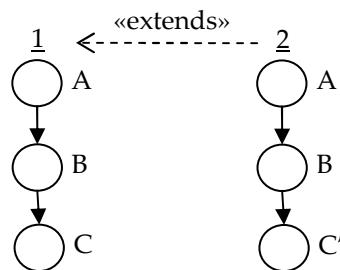
- ♦ **Estensione**

L'estensione è rappresentata invece dallo stereotipo «extends» e si rappresenta come:

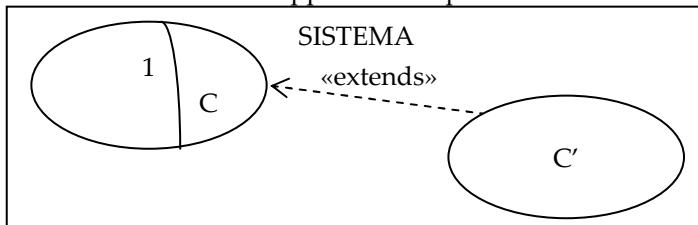


Questo tipo di associazione corrisponde alla caratteristica del polimorfismo: sono dati due oggetti diversi, i quali fanno un "qualcosa" di diverso, ma vogliono raggiungere lo stesso obiettivo.

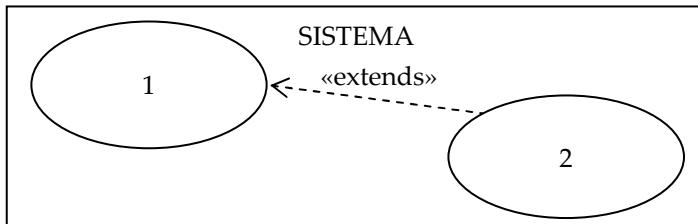
Per comprenderlo, consideriamo la seguente figura:



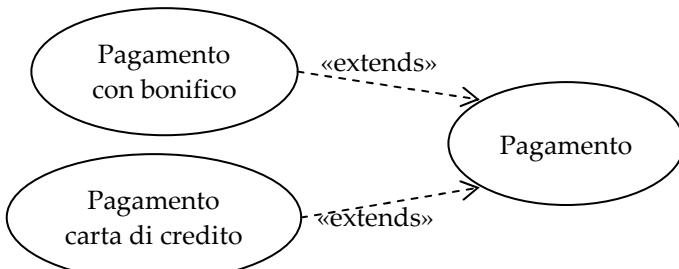
In questo caso, abbiamo un oggetto 1 e un oggetto 2, il cui comportamento è diverso. Tuttavia, entrambi hanno lo stesso obiettivo, solo che 2 lo raggiunge in maniera parzialmente diversa rispetto ad 1 (anziché eseguire C, esegue C'). Diciamo quindi (in base a come abbiamo orientato la freccia) che 2 è l'estensione di 1, mentre 1 è l'esteso. In UML dovremmo rappresentare questa situazione nella maniera seguente:



Dove 1 è il nome dello use case, mentre C è detto *punto di estensione* (cioè è l'elemento che può essere sostituito). Tuttavia, per maggiore semplicità la notazione che si utilizza è quella già introdotta in precedenza, ovvero:



In sostanza quindi si finge che il punto di estensione coincida con l'intera attività. Un esempio di applicazione è il seguente:



♦ Generalizzazione tra use case

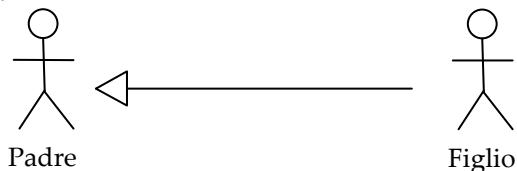
È anche possibile definire una relazione di generalizzazione tra due use case. In questo caso, lo use case figlio eredita tutti gli attributi, scenari... definiti nello use case padre (ricalca il concetto di ereditarietà della programmazione ad oggetti). Inoltre, lo use case figlio partecipa a tutte le relazioni in cui è coinvolto lo use case padre e può prevedere nuovi scenari non previsti nello use case padre.

Uno use case può ereditare da n altri use case e può essere il padre di n altri use case.

Non approfondiremo molto lo generalizzazione tra use case.

Legami tra attori: generalizzazione

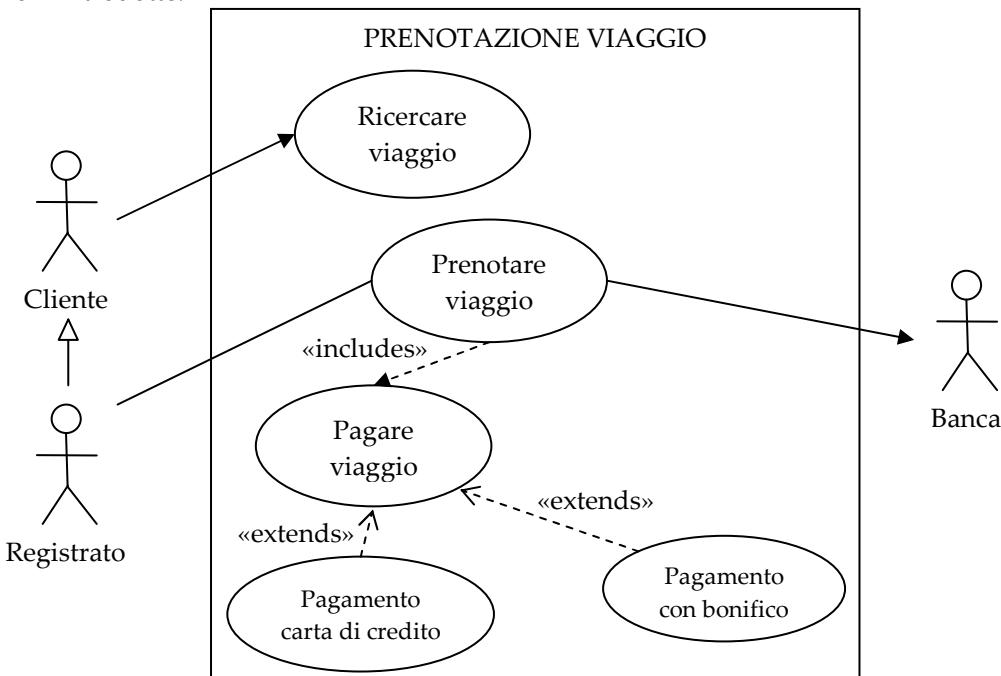
L'unico possibile legame tra due attori è quello di generalizzazione. La generalizzazione tra attori si rappresenta nel modo seguente:



L'attore figlio eredita tutte le caratteristiche dell'attore padre e può partecipare a tutti gli use case a cui partecipa l'attore padre.

Esempio di use case diagram

Possiamo completare come esempio lo use case relativo alla prenotazione di un viaggio, inserendo le nuove nozioni introdotte:



Si osserva che solitamente gli attori che usano l'applicazione sono rappresentati a sinistra del sistema, mentre quelli che vengono "usati" dal sistema sono posizionati a destra del sistema stesso.

Si mette inoltre in evidenza che non è stato indicato il legame tra l'attore "Registrato" e lo use case "Ricercare viaggio", perché siccome l'attore eredita da "Cliente", eredita anche il legame con tale use case. Inoltre, è bene osservare che solitamente il diagramma dei casi d'uso è accompagnato da una descrizione testuale (in linguaggio naturale) che consenta di capire meglio ciò che è stato rappresentato nel diagramma (ad esempio, come già accennato, nel diagramma non compare in alcun modo il tempo).

3. Activity diagram

Cosa sono?

Gli activity diagram sono molto simili ad un semplice diagramma di flusso (*flow diagram*): essi rappresentano infatti un flusso di attività (come ci suggerisce il loro nome). In particolare, tali diagrammi vengono usati per descrivere il comportamento dinamico di un sistema, fornendo la sequenza di operazioni che definiscono un'attività più complessa.

Un Activity Diagram può essere associato

1. Ad una classe.
2. All'implementazione di una operazione.
3. Ad uno Use case.

Attività

Un'attività è un lavoro svolto da un oggetto in maniera continuativa. Le attività possono essere:

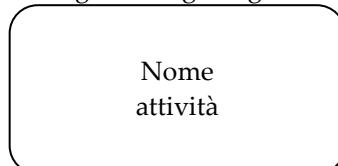
- **Attività atomiche (action state)**

Sono attività eseguibili in maniera atomica, cioè che non possono essere decomposte in una sequenza di attività, né possono essere interrotte.

- **Attività non atomiche (activity state)**

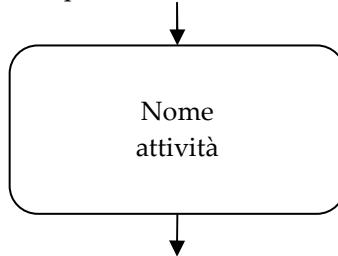
Sono attività non eseguibili in maniera atomica: possono cioè essere decomposte in una sequenza di attività e possono anche essere interrotte durante la loro esecuzione. Un'attività di questo tipo può a sua volta essere descritta mediante un activity diagram.

In entrambi i casi il simbolo usato è un rettangolo con gli angoli smussati:



L'attività ha un inizio ed una fine, ed il controllo di tutto il flusso rimane all'attività per tutto il tempo che intercorre tra l'inizio dell'attività e la sua fine (tempo che può anche essere molto lungo).

Il passaggio del controllo da un'attività ad un'altra è detto *transizione*. Ovviamente, la transizione avviene quando termina l'attività sorgente e dà il controllo all'attività di destinazione. Ogni attività ha una transizione in ingresso ed una transizione in uscita, entrambe rappresentate mediante frecce (quella in ingresso è una freccia entrante nell'attività, quella in uscita è uscente):



Se l'attività è contraddistinta da un * in alto a sinistra, significa che l'attività può essere ripetuta più volte.

Inizio e fine del diagramma

All'interno dell'activity diagram esistono anche due attività "fittizie", che sono l'inizio e la fine del diagramma stesso. L'inizio è rappresentato mediante un cerchio pieno, mentre la fine è rappresentata da un cerchio pieno all'interno di un cerchio vuoto più grande. L'attività di inizio ha solo una transizione di uscita, mentre quello d'uscita ha solo una transizione d'uscita.

In figura sono rappresentati i simboli dell'inizio (a sinistra) e della fine (a destra) del diagramma.

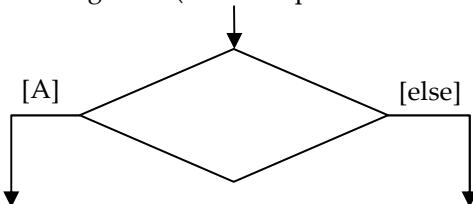


Branch (divisione) & merge (fusion)

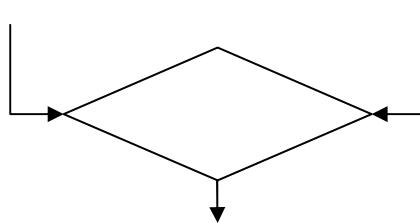
Un branch è un punto a partire dal quale il flusso di controllo si divide in due o più cammini. Si tratta quindi di un'attività particolare, con una sola transizione d'ingresso e almeno due transizioni d'uscita. Ad ogni freccia d'uscita è associata una condizione. Tali condizioni devono essere mutuamente esclusive, e si deve sempre fare in modo che esattamente una di tali condizioni sia vera. L'unica transizione che avviene veramente è quella associata alla condizione verificata in quella particolare istanza del diagramma.

È anche possibile che un ramo d'uscita non abbia ad esso associata alcuna condizione (o che vi sia scritto *else*): in tal caso, quel ramo viene eseguito quando nessuna delle altre condizioni risulta verificata.

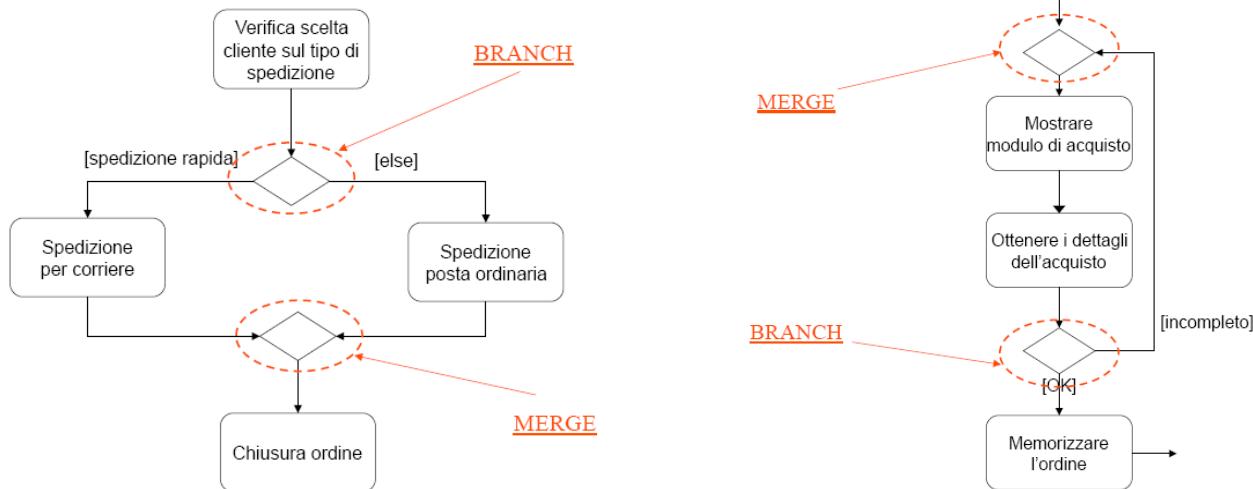
Il branch è perciò una struttura del tutto analoga ad un *if* (ad una condizionale) in un linguaggio di programmazione. Il simbolo usato è il seguente (nell'esempio abbiamo 2 cammini possibili):



Il branch deve poi essere seguito da un simbolo di fusione (merge). Il merge è il punto in cui i diversi cammini si riuniscono. In pratica, quando arriva il primo percorso in ingresso al merge, si va avanti, passando il controllo all'attività di destinazione dell'unica transizione d'uscita del merge stesso. Il simbolo è il seguente:



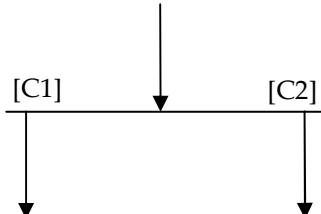
Di seguito sono riportati due esempi, il secondo dei quali mette in evidenza come sia possibile realizzare un'iterazione utilizzando il branch & merge e una semplice transizione che "ritorna indietro":



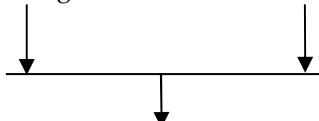
Fork & Join

Il flusso può anche essere diviso mediante un altro tipo di costrutto, il *fork*. Tale costrutto però divide il flusso di controllo in due o più flussi indipendenti (threads), che proseguono in parallelo. Tutti i flussi vengono quindi attivati (e non solo uno, come accadeva con il branch), a patto che le eventuali condizioni ad essi associate siano verificate (le condizioni non devono essere mutuamente esclusive, ma comunque devono essere costruite in modo che almeno una di tali condizioni risulti essere vera).

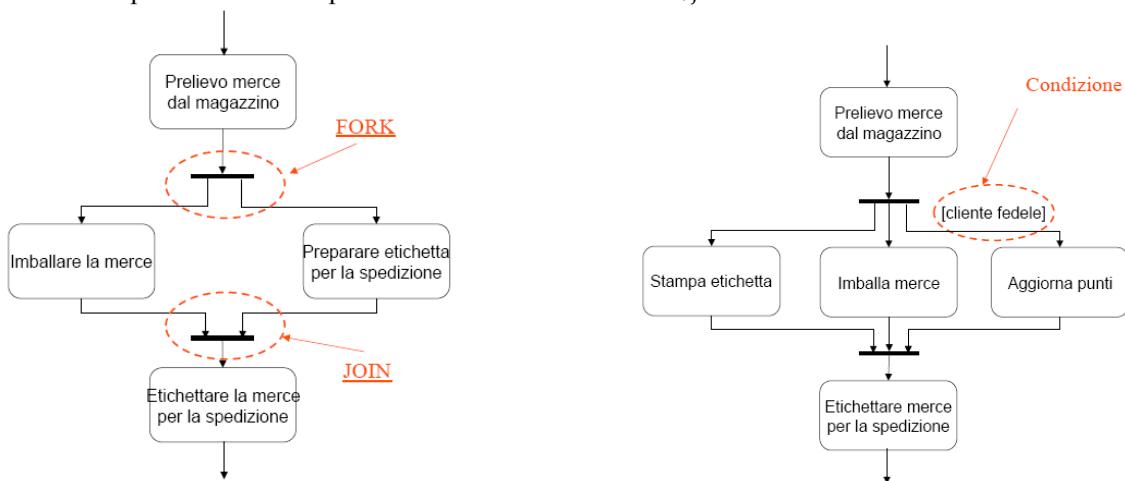
Il simbolo del fork è:



Ogni fork deve poi essere accompagnato da un simbolo di join, che rappresenta la sincronizzazione di due o più flussi in uno solo. Il join prevede quindi che si attenda il completamento di tutti i processi sulle linee di ingresso (a patto che siano stati attivati), e solo quando tutti sono giunti al simbolo di join, si procede con l'attività successiva. Il simbolo del join è il seguente:

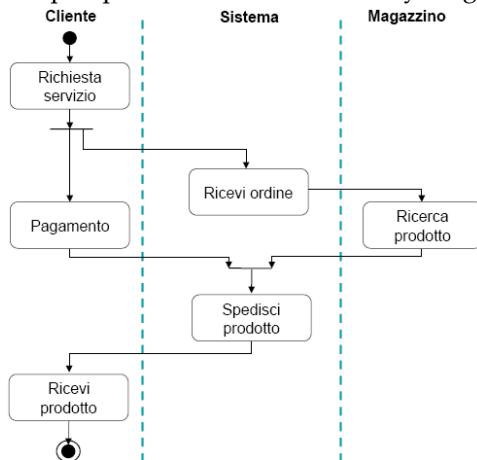


Si seguito sono riportati due esempi di uso del meccanismo fork/join:



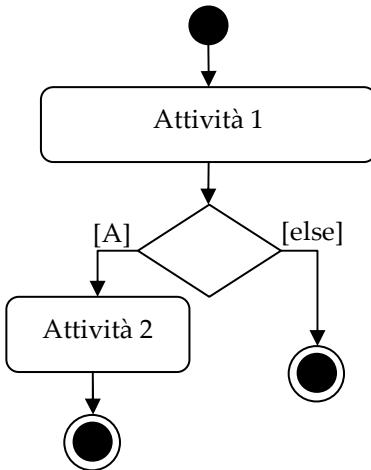
Swimlane

È possibile anche separare diverse *swimlane* in modo tale da separare le diverse responsabilità in relazione alle varie operazioni. Nei business model, che analizzeremo in seguito, ogni swimlane corrisponde ad una diversa unità organizzativa. Ad esempio, possiamo avere un activity diagram del tipo:

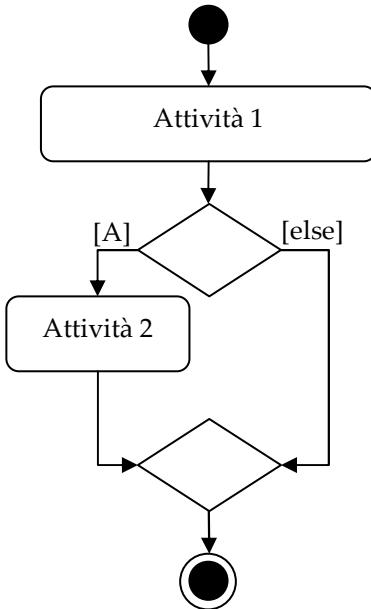


Activity diagram robusto

Un activity diagram si dice robusto se tutti i costrutti sono correttamente annidati. Ad esempio, il seguente diagramma non è robusto, anche se è corretto:



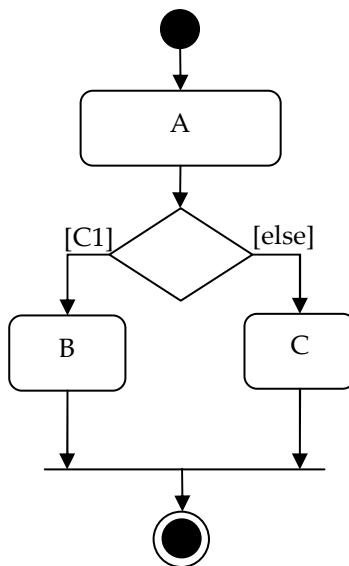
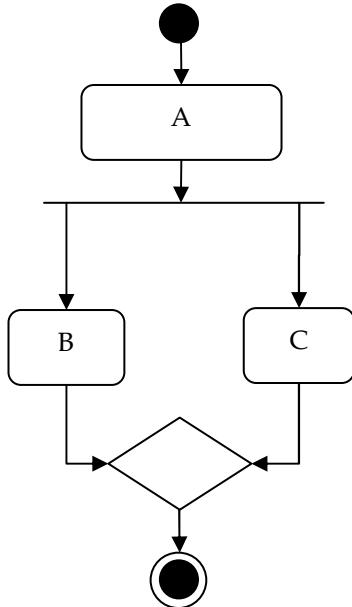
Un activity diagram robusto ha sempre uno ed un solo simbolo di fine. Il precedente diagramma dovrebbe allora diventare:



Si noti che dal punto di vista semantico, i due diagrammi sono equivalenti: l'unica differenza è che il secondo è robusto, mentre il primo non lo è.

Osservazioni sugli activity diagram e sulla divisione del flusso

Si noti che in realtà non sempre un branch è seguito da un merge, e non sempre un fork è seguito da un join: il costrutto branch/merge e il fork/join possono infatti essere “mescolati”, come mostrato nei seguenti esempi:



Nell'esempio di sinistra, le attività B e C vengono avviate contemporaneamente e poi, appena termina la prima, si termina l'intero processo. Nell'esempio di destra invece la semantica è ambigua: teoricamente infatti si dovrebbe attendere la terminazione di entrambi i rami in ingresso al join, ma questo equivale ad attendere per un tempo infinito (essendoci infatti un branch, solo uno di tali rami verrà attivato). Si potrebbe però anche voler intendere che si deve aspettare la terminazione dei soli rami attivati (e quindi in questo caso la semantica è equivalente a quella che si avrebbe con un merge al posto del join).

4. Class diagram

Che cosa sono i diagrammi delle classi?

I diagrammi delle classi rappresentano le classi e gli oggetti, con i relativi attributi ed operazioni, che compongono il sistema. Il loro obiettivo è quello di visualizzare la parte statica del sistema.

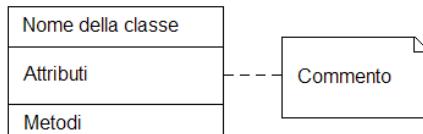
Il diagramma delle classi specifica, mediante le associazioni, i vincoli che legano tra loro le classi.

Si osserva inoltre che può essere definito a diversi livelli (analisi, disegno di dettaglio).

Rappresentazione delle classi

- ♦ *La classe*

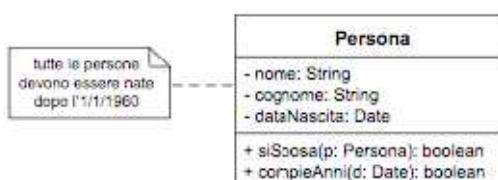
La classe viene rappresentata mediante un rettangolo, che è a sua volta suddiviso in tre parti, come mostrato in figura:



Il nome permette di identificare l'elemento del quale stiamo parlando, gli attributi consistono nell'elenco degli elementi che ci consentono di definire i singoli oggetti appartenenti a tale classe e i metodi descrivono il loro comportamento.

La rappresentazione della classe può anche essere accompagnata da un commento, riportato a lato e all'interno di un riquadro che rappresenta in maniera stilizzata un foglietto.

Ad esempio, potremo avere:



Accanto al nome della classe possiamo anche indicare in parentesi graffe le sue proprietà (ad esempio, possiamo indicare abstract se si tratta di una classe astratta).

- ♦ *Gli attributi*

Come mostrato nel precedente esempio, gli attributi vengono elencati con un particolare formato, che è il seguente:

<Visibilità> <Nome Attributo>: <Tipo Attributo> = <Valore di default> {<Stringa di proprietà>}

Dove il campo <Visibilità> può assumere i valori seguenti, con il significato riportato a lato:

| | |
|---|-------------------------------------------------|
| - | Indica che l'attributo ha visibilità private. |
| + | Indica che l'attributo ha visibilità public. |
| # | Indica che l'attributo ha visibilità protected. |
| ~ | Indica che l'attributo ha visibilità friendly. |

La stringa di proprietà può ad esempio assumere valori come changeable, froze, addonly, Il significato dei restanti campi è invece molto intuitivo. Naturalmente, se non si ha alcun valore di default, la parte ad esso relativa viene semplicemente omessa.

È invece opportuno osservare che tramite la notazione UML è anche possibile mettere in evidenza quando un attributo è costante: in tal caso infatti l'attributo è sottolineato.

- ♦ *I metodi*

Il formato attraverso il quale vengono specificati i metodi è invece il seguente:

<Visibilità> <Nome Metodo>(<Lista di Parametri>): <Tipo di ritorno> {<Stringa di proprietà>}

Dove il campo <Visibilità> può assumere gli stessi valori (e con lo stesso significato) che abbiamo già elencato quando abbiamo descritto la notazione per gli attributi. Gli altri campi hanno invece un significato intuitivo. È però bene descrivere in maniera dettagliata il formato attraverso il quale viene specificata la lista di parametri: i parametri sono separati da virgola, e per ciascuno di essi scriveremo:

<Direzione> <Nome Parametro> : <Tipo Parametro>

Il campo <Direzione> è facoltativo, e può assumere i valori in, out oppure inout (con ovvio significato).

Quando tale campo è omesso, si sottintende che il parametro sia semplicemente un parametro in input.

Le associazioni

- **Che cosa sono le associazioni**

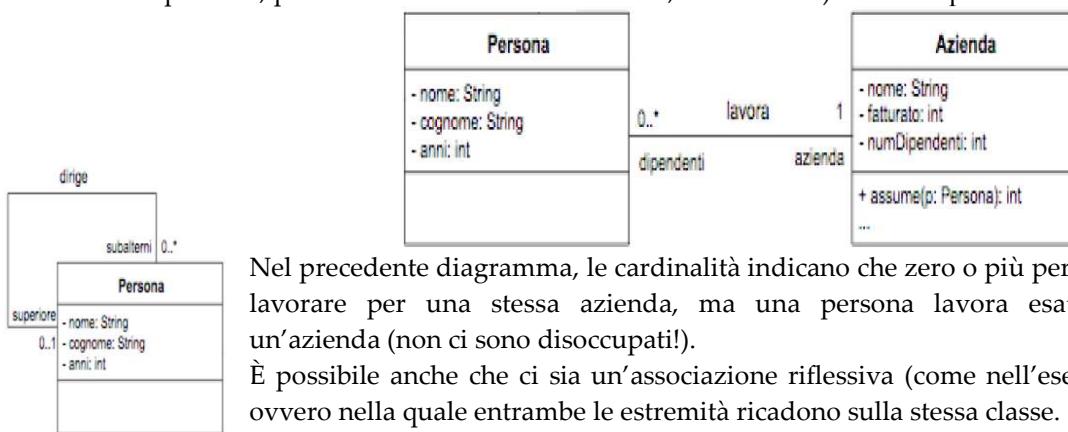
Le associazioni sono delle relazioni tra classi. Ad esempio, possiamo pensare ad una classe Persona e ad una classe Azienda: è possibile che una persona sia messa in relazione all'azienda perché lavora per essa, perciò avremo un'associazione tra Persona e Azienda.

- **Come rappresentare le associazioni**

Le associazioni vengono rappresentate mediante delle linee che congiungono le classi interessate dall'associazione stessa. Su tale linea viene inoltre indicata una parola per descrivere la tipologia di associazione (solitamente un verbo: nel precedente esempio, potremmo usare "lavorare").

Inoltre, si può opzionalmente indicare una descrizione dei ruoli svolti dalle singole classi all'interno dell'associazione, mediante un nome (come mostrato nella figura seguente, dove tali termini sono "dipendenti" e "azienda").

Gli estremi della classe prevedono che si indichi una cardinalità (o molteplicità): su ogni estremo troveremo un'indicazione del tipo: 1 (esattamente 1), oppure 0..1 (zero o 1), oppure 1..*, (uno o più), 0..* (zero, uno o più di uno), n (dove n è un numero qualsiasi, indica esattamente n), n1-n2 (dove n1 e n2 sono numeri qualsiasi, purché n2 > n1: indica minimo n1, massimo n2). Ad esempio:



Nel precedente diagramma, le cardinalità indicano che zero o più persone possono lavorare per una stessa azienda, ma una persona lavora esattamente per un'azienda (non ci sono disoccupati!).

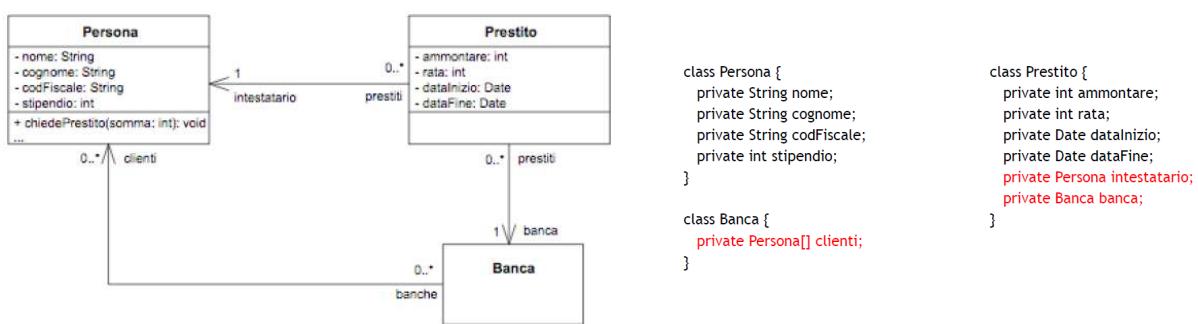
È possibile anche che ci sia un'associazione riflessiva (come nell'esempio a lato), ovvero nella quale entrambe le estremità ricadono sulla stessa classe.

- **Come si traducono le associazioni: l'uso delle frecce**

Le associazioni si traducono concretamente attraverso l'inserimento di opportuni attributi. In particolare, ogni estremo è un "attributo implicito": se la cardinalità è 0..1 oppure 1, allora l'attributo sarà semplicemente una variabile di tipo di riferimento all'altra classe interessata dall'associazione, altrimenti si tratterà di un array o di una collezione.

Tuttavia, inserire un attributo aggiuntivo da entrambe le parti è ridondante: ciò potrebbe comunque rivelarsi comodo, per facilitare ad esempio le operazioni di ricerca, ma talvolta lo si vuole evitare. In questo caso, si inserisce sulla linea che rappresenta l'associazione una freccia. La classe verso la quale è rivolta la freccia non conterrà attributi aggiuntivi per memorizzare l'associazione.

Ad esempio:



Le aggregazioni

- ♦ *Che cosa sono le aggregazioni*

Le aggregazioni sono una forma particolare di associazione; in particolare, si tratta di associazioni nelle quali una classe è in relazione con un'altra perché una di esse rappresenta una parte dell'altra. Diciamo quindi che si mettono in relazione un oggetto con una sua parte.

In sintesi quindi le aggregazioni servono per indicare quando un oggetto è costruito da altri oggetti.

- ♦ *Le composizioni*

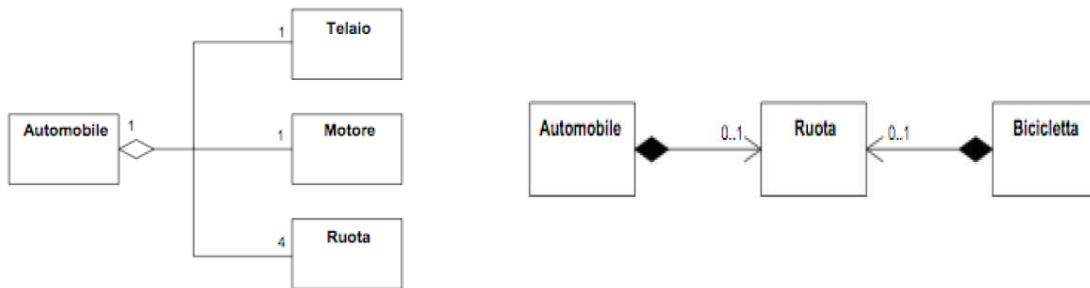
Le composizioni poi sono un caso particolare di aggregazione: una composizione è infatti un'aggregazione forte, ovvero un'aggregazione nella quale le parti componenti non esistono senza il contenitore. Ciò significa che la creazione e la distruzione dei componenti avvengono all'interno del contenitore e che i singoli componenti non possono essere parti di altri oggetti.

In Java le composizioni di fatto non vengono usate. In ogni caso, aggregazioni e composizioni vengono poi tradotte allo stesso modo al momento di stendere il codice.

- ♦ *Rappresentazione di aggregazioni e composizioni*

Le aggregazioni e le composizioni vengono rappresentate mediate delle linee (come le normali associazioni), alle quali però non è associato un verbo (perché è scontato il tipo di relazione tra le due classi); si utilizza però un rombo in prossimità della classe che rappresenta l'oggetto completo. Si indicano comunque le cardinalità, come nelle normali associazioni.

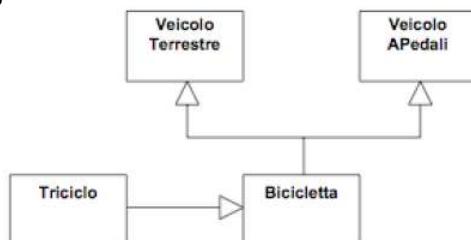
La distinzione tra normale aggregazione e composizione è legata al colore del rombo: se il rombo è bianco, si tratta di un'aggregazione, se è nero, si tratta di una composizione. Di seguito sono riportati due esempi, il primo è un'aggregazione, il secondo è una composizione.



L'ereditarietà e le interfacce

- ♦ *Come si rappresenta l'ereditarietà*

In UML l'ereditarietà assume il nome di *generalizzazione*. La generalizzazione si rappresenta in UML come mostrato nell'esempio seguente:



Dove si indica che Triciclo eredita da Bicicletta e che Bicicletta eredita da VeicoloTerrestre e da VeicoloAPedali. Sappiamo che quest'ultima situazione in Java non può esistere, perché non si ha ereditarietà multipla; occorre tuttavia ricordare che UML non è legato a Java e che in altri linguaggi di programmazione è possibile realizzare l'ereditarietà multipla.

Le classi che ereditano da un'altra, naturalmente, vedranno indicati solamente metodi ed attributi aggiuntivi e non quelli ereditati dalla superclasse (in quanto sono impliciti).

- ♦ **Le interfacce**

Le interfacce in UML possono essere rappresentate in due diversi modi. Il primo modo prevede semplicemente che si usi la stessa rappresentazione adottata per le classi, ma escludendo la parte relativa agli attributi e aggiungendo la scritta <<interface>> appena prima del nome, come nel seguente esempio:



Come mostrato nell'esempio, l'implementazione di una classe è indicata così come la generalizzazione (ereditarietà), ma con linea tratteggiata. Analogamente, le associazioni che coinvolgono l'interfaccia (come quella tra FiguraGeometrica e List) sono indicate con una linea tratteggiata.

Esiste poi una seconda rappresentazione semplificata, che è la seguente:



Tale rappresentazione però è meno chiara, perché non indica esplicitamente i metodi definiti nell'interfaccia.

L'ereditarietà e le interfacce

- ♦ **I package**

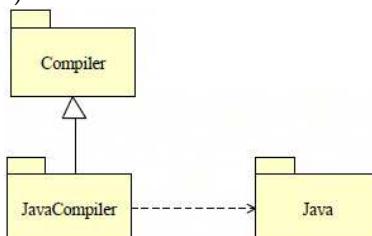
In UML, così come in Java, esiste il concetto di package. Un package è un meccanismo generale per organizzare degli elementi in gruppi omogenei.

Un package può contenere altri package, e inoltre possono esserci delle relazioni tra i vari package.

Si osserva però che il concetto di package in UML è diverso da quello in Java: in UML possiamo infatti instaurare anche relazioni di dipendenza e di generalizzazione (ereditarietà) tra package. Ad

- ♦ **Rappresentazione dei package e delle loro interazioni**

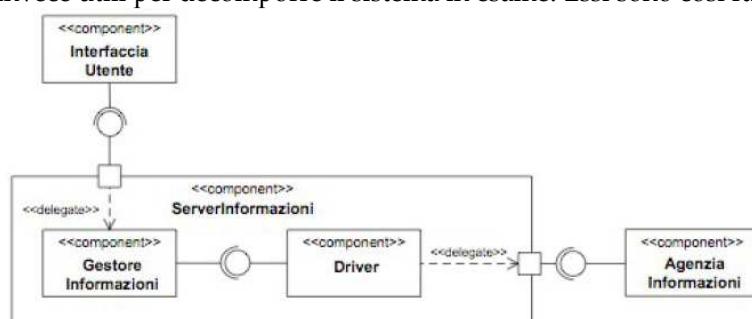
Di seguito è riportato un esempio di rappresentazione di package, dove è mostrata anche una relazione di generalizzazione e una diversa relazione di dipendenza (tipicamente ciò significa che il package JavaCompiler accede al package Java).



Componenti

- ♦ **I componenti**

I componenti sono invece utili per decomporre il sistema in esame. Essi sono così rappresentati:



I "quadrati" ai margini del componente rappresentano delle porte, che implementano delle interfacce e dei protocolli opportuni per la comunicazione con l'esterno.

1. Diagrammi di sequenza

I diagrammi di sequenza

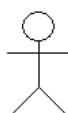
I diagrammi di sequenza consentono di rappresentare l'interazione tra oggetti, materializzando così degli scenari specifici. In particolare, essi evidenziano il modo in uno scenario (ovvero uno specifico percorso in un caso d'uso) viene risolto dalla collaborazione tra un insieme di oggetti. Per fare, cioè, specifica la sequenza dei messaggi scambiati tra gli oggetti.

I diagrammi di sequenza possono specificare nodi decisionali e iterazioni. Tali diagrammi sono perciò utili per due motivi:

1. Per evidenziare le interazioni tra gli oggetti e quindi i metodi da associare alle diverse classi;
2. Per provare l'efficacia dei metodi identificati.

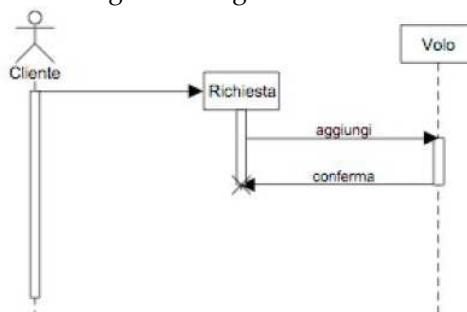
Come si realizzano i diagrammi di sequenza

I diagrammi di sequenza prevedono che ci sia una "linea verticale", che rappresenta la linea del tempo, mentre in orizzontale sono disposti i vari oggetti. Oltre agli oggetti, viene rappresentato anche l'utente (il cliente), mediante il disegno seguente, che risulta molto intuitivo.

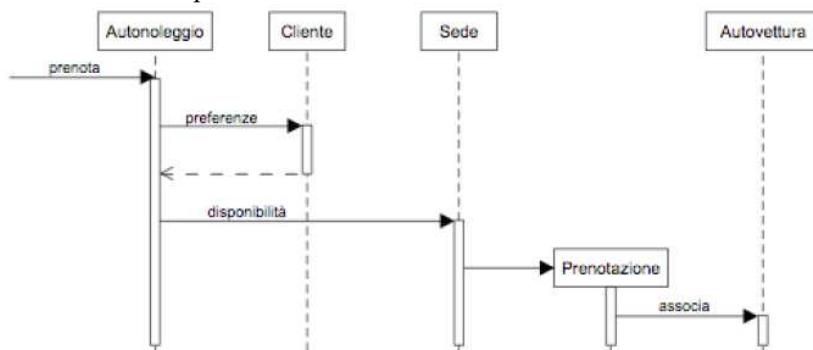


Quando ad un oggetto corrisponde una linea verticale tratteggiata, significa che quell'oggetto esiste, ma è passivo (cioè non fa nulla); se invece la linea verticale è doppia, significa che l'oggetto è attivo. Infine, se la linea non è presente, significa che l'oggetto non è ancora stato creato, oppure è stato distrutto (sappiamo che in Java gli oggetti non vengono realmente distrutti, perché ci penserà poi il garbage collector, però possiamo considerare tale istante, a livello logico, come l'istante in cui si perde il riferimento a quell'oggetto).

Le interazioni tra oggetti vengono rappresentate mediante frecce, sulle quali è riportato un nome o un verbo che specifica il tipo di interazione. Ad esempio, se vogliamo rappresentare la sequenze per la prenotazione di un volo, disegneremo il seguente diagramma:



Di seguito è riportato un altro esempio:

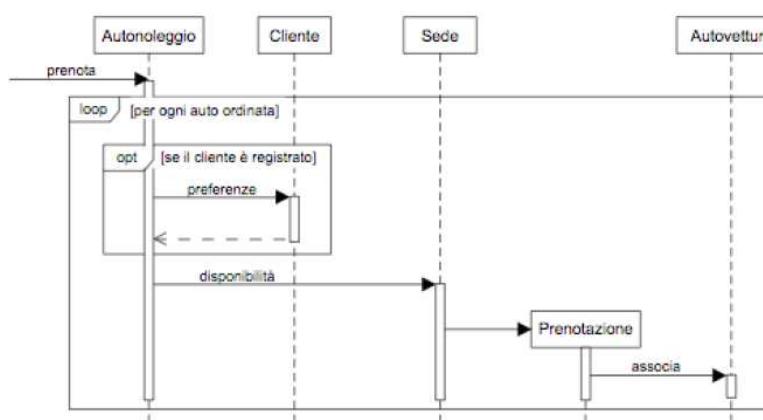


Potrebbe però rivelarsi utile anche definire dei cicli, oppure delle alternative, Per tale scopo sono stati introdotti i frame d'interazione, che consentono appunto di specificare questo tipo di situazioni.

Alcuni dei più usati frame di interazione sono i seguenti:

| | |
|------|----------------------------------------------------------------------------------------------------------------------------------------------|
| alt | Indica un'alternativa del tipo (o ... o): se è verificata una condizione si fa una certa cosa, altrimenti se ne fa un'altra. |
| opt | Indica un'azione opzionale, che viene cioè svolta solo se è verificata una certa condizione (in caso contrario non si fa nulla). |
| loop | Indica l'iterazione di una certa operazione. La condizione di iterazione è espressa da una descrizione a parole affiancata alla parola loop. |
| ref | Indica un'operazione che è descritta all'interno di un diverso diagramma. |
| par | Indica che più frammenti vengono eseguiti in parallelo. |
| neg | Serve per evidenziare un'interazione non valida. |

All'interno di ogni frame si specifica poi a parole una breve descrizione che indica la condizione entro la quale certe operazioni vengono eseguite e/o iterate. Ad esempio, possiamo usare i frame di interazione per correggere il precedente esempio in modo tale che si eviti che nell'intervallo tra la verifica di disponibilità e l'effettiva prenotazione, la disponibilità dell'auto venga a mancare: tale situazione creerebbe ovviamente un errore. Possiamo allora disegnare il seguente diagramma di sequenza:



Capitolo 7: BPMN

1. BPMN

Che cos'è BPMN

Il BPMN (Business Process Modeling Notation) è una notazione grafica definita da OMG (Object Management Group) per la modellazione dei processi di business. Si ricorda che un processo di business è un insieme di attività strutturate in un certo modo, che producono e/o usano informazioni allo scopo di arrivare a fornire un prodotto o un servizio finale.

Uno schema BPMN è simile ad un activity diagram di UML, tuttavia la sintassi messa a disposizione da BPMN è più ricca.

2. Pool e lane

I pool

Lo schema BPMN è sempre suddiviso in pool. Ogni pool identifica a livello logico una diversa organizzazione, indipendente dalle altre, ma interagente con esse. Possiamo anche dire che il pool corrisponde ad uno specifico ruolo (dove allo stesso ruolo possono essere associate più persone fisiche).

I pool sono rappresentati mediante dei rettangoli, che si estendono orizzontalmente o verticalmente, presentano uno nome e contengono una sequenza di attività e di altri vari elementi dei diagrammi BPMN, che ora andremo ad analizzare.

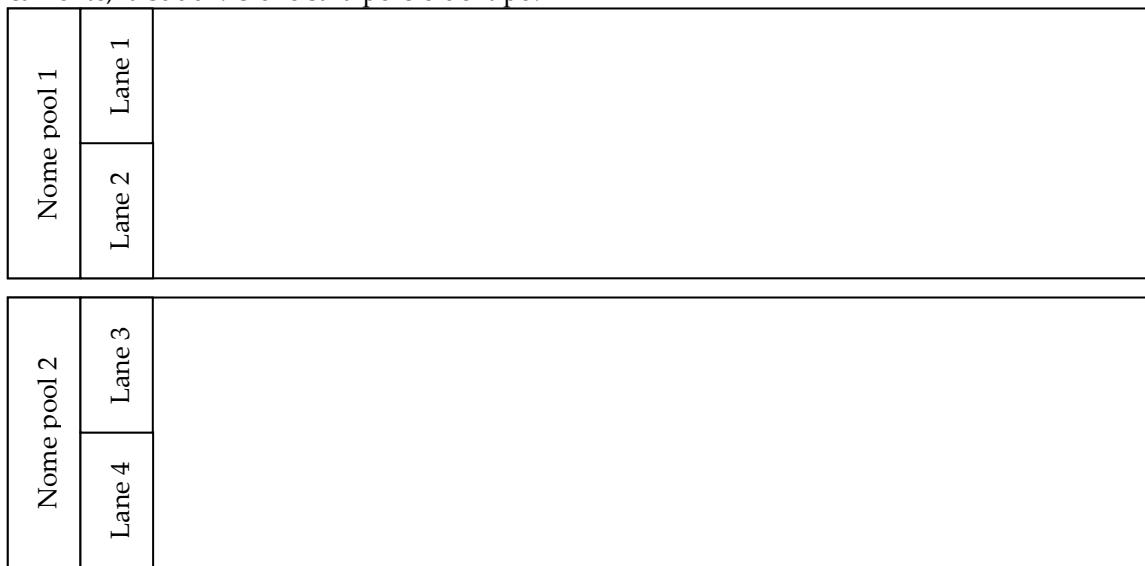
Le lane (o swimlane)

Ogni pool può poi essere suddiviso in lane (o swimlane), che rappresentano delle sotto-organizzazioni interne allo stesso ruolo. È anche possibile avere delle lane innestate.

Da un punto di vista grafico, le lane sono rappresentate suddividendo in varie "corsie" il pool e indicando il titolo in un rettangolo all'interno della lane, come mostrato in figura.

Rappresentazione grafica e osservazioni

Graficamente, la suddivisione sarà perciò del tipo:



È opportuno che tra due pool diversi si abbia un accoppiamento debole (*loosely coupling*), mentre l'accoppiamento tra due lane diverse dello stesso pool deve essere forte.

3. Le attività

Cosa e come si rappresentano?

- ◆ *Le attività*

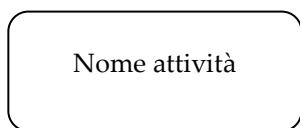
Un'attività può essere:

- a) Un'attività atomica, ovvero un singolo task, non ulteriormente scomponibile, eseguito da uno degli attori del sistema.
- b) Un'attività composta, ovvero un insieme di attività che può poi essere a sua volta scomposto in un diagramma BPMN. Di conseguenza, un'attività composta contiene un "sotto-processo", che può essere descritto in uno schema BPMN a parte: quando si esegue l'attività composta, si passa di fatto ad eseguire il BPMN ad essa associata, e quando si arriva al punto di fine di tale schema si riparte con l'esecuzione dello schema al quale appartiene l'attività composta, dall'attività che segue quest'ultima.

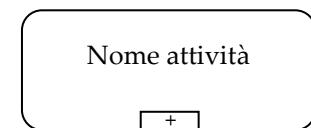
- ◆ *Rappresentazione delle attività*

Le attività sono rappresentate graficamente mediante un rettangolo con gli angoli smussati, contenente il nome dell'attività. Se l'attività è composta, essa presenta inoltre un simbolo "+".

Attività atomica



Attività composta



Tipi di attività

Le attività possono essere di 7 tipi (ma vengono rappresentate tutte allo stesso modo):

- ◆ *Service task*

È un servizio eseguibile da un servizio software su cui non si può agire (un pezzo di software).

- ◆ *Receive task*

È un task che, per poter essere eseguito, deve attendere la ricezione di un messaggio.

- ◆ *Send task*

È un task che, al termine della propria esecuzione, spedisce un messaggio in uscita.

- ◆ *User task*

È un task nella cui esecuzione entra in gioco l'utente (essere umano): il sistema informa l'utente circa un'attività da svolgere.

- ◆ *Manual task*

L'utente esegue il task senza alcun componente software e informa il sistema quando l'attività è finita.

- ◆ *Script task*

Si ha una funzione che esegue le attività in modalità batch.

- ◆ *Reference task*

Si tratta di task usati per indicare che due attività sono uguali tra loro. Nel nostro caso, indicheremo questa caratteristica semplicemente attribuendo a tali attività lo stesso nome.

Marcatura delle attività per indicare le iterazioni

Un'attività può poi avere alcune particolari caratteristiche, che possono essere definite mediante marcature:

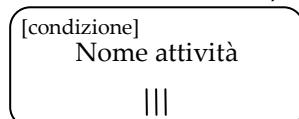
- ♦ **Attività ciclica con iterazioni sequenziali (standard loop)**

Un'attività che può essere svolta una o più volte, in maniera ciclica e sequenziale, è identificata dal marcitore di standard loop, come mostrato in figura. A destra è riportato anche un esempio di uso.



- ♦ **Attività ciclica con iterazioni in parallelo (multiple instance loop)**

Se invece è possibile che l'attività venga eseguita più volte, ma le sue singole esecuzioni possono avvenire parallelamente tra loro, il simbolo usato è detto *multiple instance loop*:



In questo caso quindi la seconda iterazione non dipenderà dal risultato della prima.

4. Sequence flow e message flow

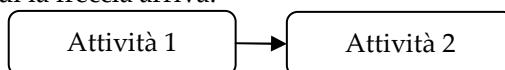
I sequence flow

- ♦ **I sequence flow**

Un sequence flow è un collegamento tra due attività all'interno di uno stesso pool. Il sequence flow in particolare consente di indicare il flusso di sequenza tra varie attività: esso indica infatti quale attività deve essere eseguita al termine dell'esecuzione di un'altra attività.

- ♦ **Simbolo**

Il sequence flow è semplicemente rappresentato mediante una linea orientata continua (una freccia con tratto continuo): si indica in questo modo che, terminata l'esecuzione dell'attività da cui parte la freccia, si passa ad eseguire quella a cui la freccia arriva.



- ♦ **Regola generale**

Un sequence flow può collegare tra loro due attività di uno stesso pool (eventualmente anche all'interno di lane diverse), ma mai due attività appartenenti a due pool diversi.

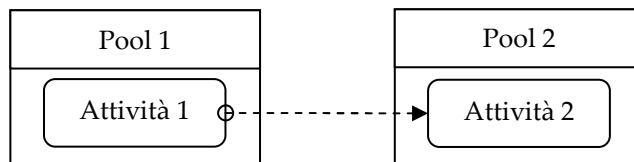
I message flow

- ♦ **I message flow**

Un message flow è un collegamento tra due attività appartenenti a pool diversi. Si ha perciò un'attività all'interno di un pool (detta *send task*) che invia un messaggio ad un'altra (*receive task*), la quale prima di poter essere eseguita deve attendere l'arrivo di un messaggio.

- ♦ **Simbolo**

Il message flow è rappresentato mediante una linea orientata tratteggiata, la cui origine è evidenziata mediante un pallino: si indica in questo modo che l'attività da cui parte la freccia invia un messaggio all'attività cui la freccia arriva, la quale quindi viene attivata al termine dell'esecuzione della precedente.



- ♦ **Regola generale**

Un message flow può collegare tra loro due attività di pool diversi, ma mai due attività appartenenti allo stesso pool (nemmeno se su lane diverse).

5. Attività iniziali, attività finali e multi-istanza

Attività iniziale (start activity)

- ◆ **Che cos'è**

L'attività iniziale è la prima attività che costituisce l'esecuzione di un processo. All'interno di un diagramma è possibile avere più punti di start, ma non si possono avere più punti di start nello stesso pool.

- ◆ **Come viene individuata?**

L'attività iniziale può essere individuata semplicemente dal fatto che non ha sequence flow o message flow entranti al suo interno. In alternativa, è possibile indicare la start activity mediante un cerchio con bordo sottile:



Tale simbolo, come vedremo a breve, indica in realtà un evento (in questo caso, l'evento di inizio) e può contenere al suo interno altri simboli che specificino meglio la natura dell'evento che determina l'avvio del processo.

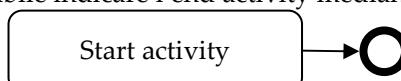
Attività finale (end activity)

- ◆ **Che cos'è**

L'attività finale è l'ultima attività che costituisce l'esecuzione di un processo.

- ◆ **Come viene individuata?**

L'attività finale può essere individuata semplicemente dal fatto che non ha sequence flow o message flow uscenti. In alternativa, è possibile indicare l'end activity mediante un cerchio con bordo spesso:



Anche in questo caso il simbolo indica un evento e potrà contenere altri particolari simboli.

Osservazione sul numero di istanze

Si noti che un'attività può anche avere un numero di sequence flow in ingresso superiore ad 1. In tal caso, l'attività viene attivata ogni volta che viene attivato uno dei percorsi entranti.

È possibile anche che un'attività abbia più di un sequence flow in uscita. In tal caso, al termine dell'attività partiranno in parallelo tutte le linee d'uscita dell'attività. In tal caso, l'engine che esegue il processo avvierà contemporaneamente diverse istanze dello stesso processo.

Per gestire l'esecuzione multi-istanza, all'evento di start viene generato un token, che deve attraversare tutto il processo. Se si ha una sola istanza, allora il token rimane unico. Le attività che hanno un solo sequence flow in ingresso e più sequence flow in uscita (*uncontrolled flow*) generano invece dei nuovi token.

6. Eventi

Eventi di inizio, di fine ed eventi intermedi

Si è già accennato alla possibilità di utilizzare all'interno di uno schema BPMN il concetto di *evento*. Gli eventi possono essere:

- ♦ **Eventi di inizio**

Sono eventi che, quando vengono intercettati (*catching*), determinano l'avvio dell'esecuzione del processo. Sono rappresentati da un cerchio con contorno sottile.

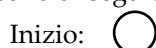
- ♦ **Eventi di fine**

Sono eventi che vengono sollevati (*throwing*) quando termina l'esecuzione del processo, fornendo così un output in termini di generazione dell'evento. Sono rappresentati da un cerchio con contorno spesso.

- ♦ **Eventi intermedi**

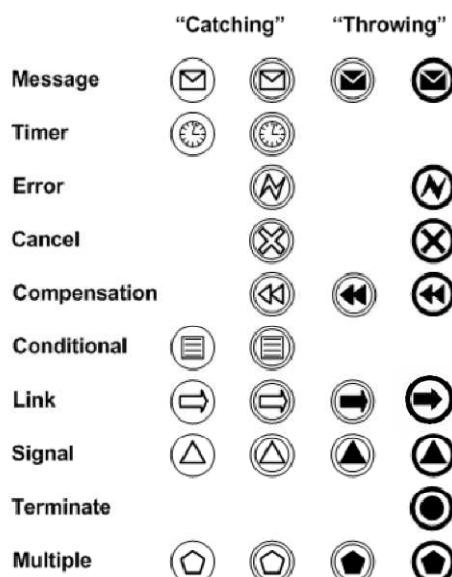
Sono eventi che vengono sollevati (*throwing*) oppure intercettati (*catching*) durante l'esecuzione del processo. Sono rappresentati da un cerchio con un doppio contorno sottile.

I simboli sono di seguito riassunti:

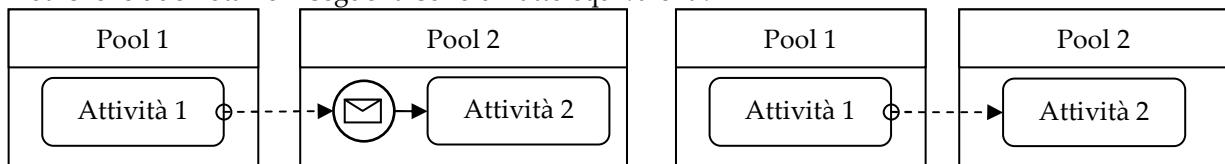


Tutti gli eventi possono avere al loro interno un particolare simbolo che ne indica la tipologia. Se il simbolo è disegnato "vuoto" (cioè sono disegnati i bordi in nero, ma il riempimento è in bianco), allora significa che l'evento viene intercettato; se invece il simbolo è rappresentato con riempimento in nero, allora si sta indicando che in quel punto viene sollevato un evento. La distinzione è importante soprattutto per gli eventi intermedi, che possono essere di entrambe le tipologie.

Lo schema seguente riassume i tipi di eventi, che verranno a breve descritti nel dettaglio.



Si noti che le due notazioni seguenti sono di fatto equivalenti:



Tipi di eventi

- ◆ **Evento generico**

Se il cerchio corrispondente all'evento viene lasciato vuoto, significa che l'evento è di tipo generico.



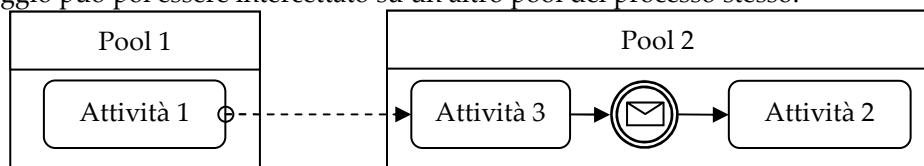
- ◆ **Evento message**

L'evento message è indicato con il simbolo di una busta.



Può essere usato:

- *Come evento di inizio*: se viene usato come evento di inizio, significa che il processo inizia solo quando arriva un messaggio (ad esempio, una telefonata, una e-mail, ...).
- *Come evento di fine*: se viene indicato come evento di fine, significa che quando il processo termina, viene inviato un messaggio.
- *Come evento intermedio*: infine, è possibile che si mandi un messaggio nel mezzo di un processo, e tale messaggio può poi essere intercettato su un altro pool del processo stesso:



In questo modo quindi si ha un meccanismo di sincronizzazione: nel pool 2, dopo aver eseguito l'attività 3, se non è già arrivato il messaggio, allora ci si pone in attesa del messaggio proveniente dall'attività 1, e solo a quel punto si passa ad eseguire l'attività 2.

- ◆ **Evento timer**

L'evento timer, indicato con il simbolo di un orologio, può essere solo intercettato e non lanciato (quindi non può essere usato come evento di fine). Esso indica la scadenza di un certo timeout. Affianco al simbolo dell'evento si deve indicare l'istante al quale "scade il tempo" (ad esempio, una data, oppure un'indicazione del tipo "tra 3 giorni", ...).



- ◆ **Evento error**

Questo evento può essere sollevato alla fine dell'esecuzione del processo, oppure può essere intercettato come evento intermedio. Viene sollevato per indicare che l'esecuzione è terminata in modo non corretto (si sta in sostanza sollevando un'eccezione).



- ◆ **Evento cancel**

Questo evento può essere sollevato alla fine dell'esecuzione del processo, oppure può essere intercettato come evento intermedio. Viene sollevato quando il processo viene annullato.



- ◆ **Evento compensation**

In alcuni casi, il processo identifica una transazione. Come noto, la transazione deve essere atomica e perciò se si decide di annullarla, si devono mettere in atto delle operazioni finalizzate ad ottenere il rollback. In un processo di business, non sempre è possibile tornare esattamente alla situazione iniziale. In ogni caso, per provare a mettere in atto meccanismi tale genere, si solleva e si intercetta l'evento di compensazione.



- ♦ **Evento conditional**

L'evento conditional, può essere solo intercettato e non lanciato. In particolare, viene sollevato quando si verifica una certa condizione, indicata accanto al simbolo dell'evento (ad esempio, si potrà avere una conduzione del tipo "estrattoConto > 2000 €").



- ♦ **Evento link**

L'evento link viene utilizzato semplicemente per "saltare" da una parte ad un'altra: consente cioè di spezzettare lo schema BPMN, al solo fine di migliorare la rappresentazione grafica. Quando si solleva il link, si indica accanto al simbolo dell'evento sollevato, anche un'etichetta, che dovrà poi essere ripetuta laddove si ha il punto in cui viene intercettato l'evento, in modo da creare un'associazione 1 ad 1 (utile soprattutto se si hanno più link nello stesso schema).



- ♦ **Evento signal**

Quando questo evento viene sollevato, si manda in broadcast a tutto il sistema una certa informazione. In tal caso, il simbolo è identificato da un triangolo colorato di nero. Il punto che intercetta tale evento è un triangolo bianco. È perciò analogo al message, ma in questo caso il messaggio non è rivolto ad un singolo destinatario, bensì viene trasmesso in broadcast.



- ♦ **Evento terminate**

L'evento terminate determina lo stop di tutti i processi in corso.



- ♦ **Evento multiple**

Il *multiple event* viene utilizzato qualora sia necessario specificare una combinazione di varie tipologie dei precedenti eventi, che si dovranno verificare contemporaneamente.

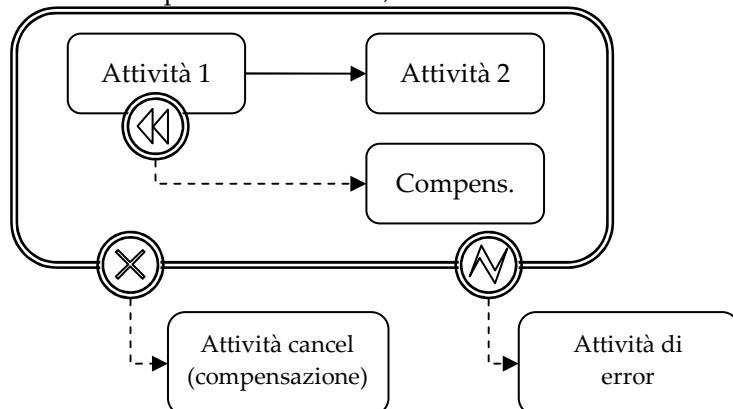


7. Transizioni

Le transizioni

Una transizione è un'attività particolare, che gode delle proprietà ACIDe. In un diagramma BPNM, le transizioni sono rappresentate mediante dei rettangoli con il bordo doppio. Al di sotto di tali rettangoli vengono inoltre indicati i simboli di sollevamento degli eventi intermedi di tipo cancel ed error, e al suo interno deve essere possibile gestire la compensazione.

In sostanza quindi, è possibile che si verifichino delle condizioni per le quali si rende necessario eseguire la compensazione. Inoltre, possono verificarsi degli errori, in corrispondenza dei quali viene lanciata l'attività di cancel, la quale tenterà di ritornare allo stato iniziale, compensando tutte le attività svolte fino a quel momento. Se anche l'attività di compensazione fallisce, allora si va in stato di errore.



8. Gateway

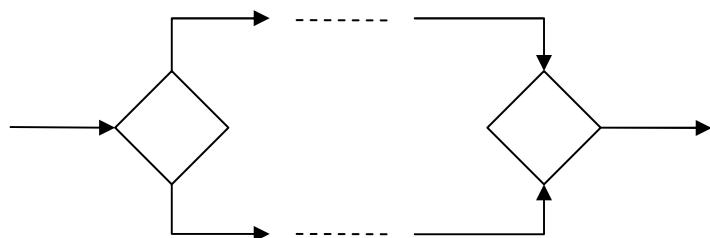
Cosa è un gateway e come si rappresenta?

- ◆ **Il gateway**

Un gateway è un punto nel quale il sequence flow si dirama in due o più percorsi (o, viceversa, due o più percorsi si riuniscono in un unico percorso, nel punto detto di *merge*).

- ◆ **Simbolo**

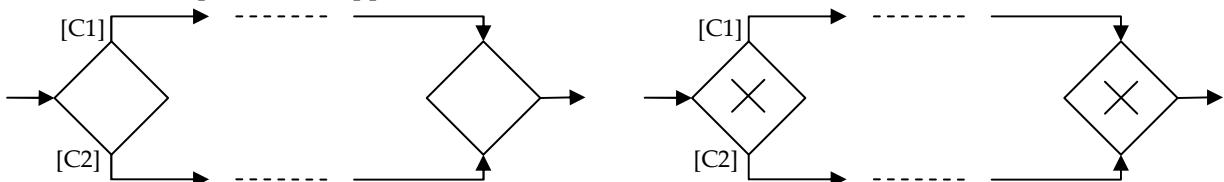
I gateway possono essere di diverso tipo tra di loro. Tutti però sono rappresentati mediante un rombo. La differenza è rappresentata dal simbolo che viene posizionato all'interno del rombo stesso.



Tipi di gateway

- ◆ **Data based exclusive choice**

Il data based exclusive choice prevede che si abbiano condizioni di uscita mutualmente esclusive e basate sui dati usati dal processo: dei percorsi di uscita ne verrà perciò attivato solamente uno. Se all'interno del rombo non si indica nulla, si intende che il gateway è di questo tipo. Inoltre, il data based exclusive choice può essere rappresentato indicando una croce all'interno del rombo (Xor):



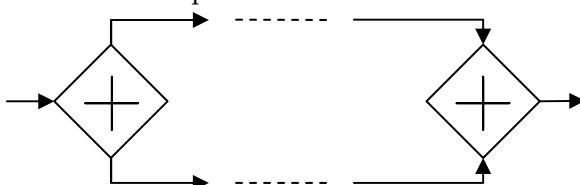
- ◆ **Data based or**

Per rappresentare in maniera esplicita la generazione di nuovi token, anziché usare un *uncontrolled flow*, si usa un opportuno gateway (*data based or*), nel quale si indica esplicitamente l'eventuale generazione di nuove istanze. In particolare, tale gateway presenta delle condizioni sui rami d'uscita, e attiva tutti e soli i percorsi le cui condizioni sono vere. È indispensabile che almeno una di tali condizioni sia valida, ma in questo caso non si deve avere mutua esclusività delle condizioni. Il merge attenderà poi la terminazione dell'esecuzione di tutti i percorsi che sono stati attivati, e solo a quel punto il sequence flow riprenderà.



- ◆ **Gateway and**

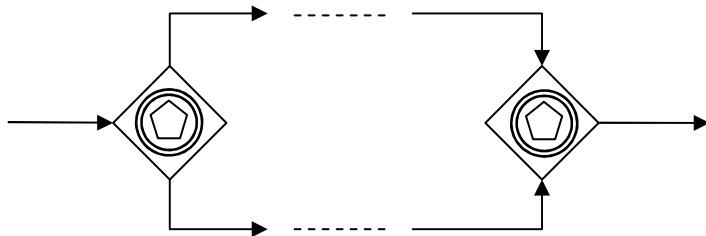
Se invece si vuole utilizzare un gateway nel quale vengano attivati contemporaneamente tutti i percorsi di uscita, si utilizza un gateway and. In questo caso non si ha alcuna condizione e il simbolo usato è il +. Si noti che in equivale al data based or nel quale tutte le condizioni sono sempre vere.



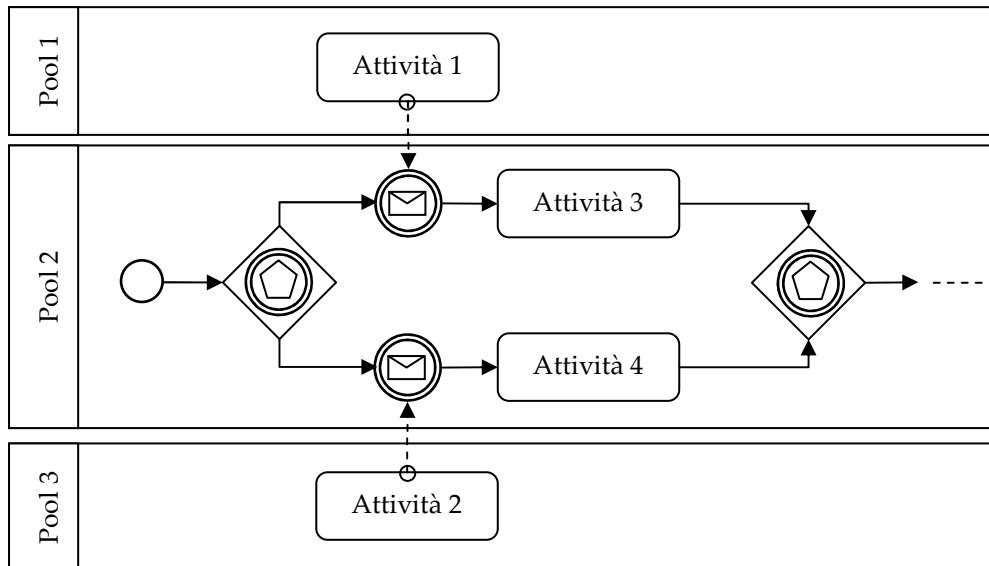
Anche in questo caso, il merge attenderà il completamento di tutti i percorsi (esegue perciò un'operazione di sincronizzazione).

- ♦ ***Event driven gateway***

Questo tipo di gateway prevede che si decida, sulla base di un evento, quale percorso seguire. Il suo simbolo è il seguente:



Ad esempio, se scriviamo:



In questo modo diciamo che, se il messaggio ricevuto è quello che ha inviato l'attività 1, allora si esegue l'attività 3; se invece il messaggio ricevuto è quello che è stato inviato dall'attività 2, si esegue l'attività 4.

Capitolo 8: Sistemi Informativi basati sul web

1. Introduzione ai WIS

Introduzione

Per rispondere alle sfide dell'economia globale, è oggi necessario valorizzare l'informazione e gli strumenti per la sua gestione: frequentemente infatti un sistema informativo deve gestire informazioni che hanno formati diversi tra loro, oppure provengono da varie fonti dislocate sul territorio (si pensi ad esempio a una multinazionale).

Naturalmente, nel raggiungimento di tale obiettivo bisogna scontrarsi con la presenza di una vasta molteplicità di tecnologie, protocolli di comunicazione, basi di dati, ecc.; di conseguenza, progettare e mantenere un sistema informativo in grado di gestire efficacemente informazioni dislocate e diverse tra loro avrà un'elevata complessità.

D'ora in avanti, ci concentreremo solamente sullo studio di un particolare tipo di sistemi informativi: i sistemi informativi basati sul Web (WIS), che hanno come caratteristica fondamentale l'uso delle tecnologia di Internet e del Web per la gestione delle informazioni. Questa scelta è proprio dovuta al fatto che le tecnologie su cui si basano i WIS sono quelle che meglio consentono di gestire informazioni miste.

I WIS sono quindi sistemi distribuiti su vari nodi, appartenenti a unità organizzative diverse che necessitano di programmi che cooperino tra loro svolgendo parti di un flusso di lavoro comune: per questa ragione i WIS rientrano nella categoria dei *sistemi cooperative*.

Definizione

Sulla base di quanto abbiamo detto nel precedente paragrafo, possiamo ora dare una definizione più precisa e formale dei WIS:

- ♦ **Definizione**

Un sistema informativo basato sul Web (WIS) è un insieme di applicazioni che, usando il Web come canale di comunicazione, sono in grado:

1. di cooperare tra loro e
2. di reperire e fornire informazioni.

- ♦ **Osservazione**

Un WIS consente di integrare molte delle funzionalità dei sistemi informativi tradizionali. Inoltre, i WIS si prestano ad integrarsi anche con i sistemi informativi già esistenti, eventualmente di altre tipologie.

- ♦ **Aspetti tecnologici di base**

Il modello di comunicazione sul quale si basa un WIS è quello tradizionale del Web, ovvero il modello client-server. Il protocollo principale per la comunicazione tra il client e il server è il protocollo HTTP, mentre il linguaggio usato per la visualizzazione delle informazioni è HTML.

Differenza tra un WIS e un normale sito Web

Come è chiaro dalla precedente definizione, il concetto di WIS è molto simile a quelli di *sito Web*; tuttavia, esistono alcune importanti differenze rispetto ai tradizionali siti Web.

In particolare, possiamo considerare i WIS come l'evoluzione naturale dei siti Web statici; tale evoluzione è caratterizzata soprattutto da una maggiore interazione con l'utente. In altri termini, anziché avere un flusso di informazioni unidirezionale verso l'utente, si ha un flusso bidirezionale, in cui le informazioni che l'utente riceve dipendono anche dalle informazioni che l'utente stesso ha inviato. In questo modo, il WIS è in grado di fornire anche dei servizi, e non solo di fornire delle informazioni statiche.

Classificazione dei sistemi sul Web

Possiamo a questo punto ulteriormente approfondire la distinzione in vari “livelli di dinamicità” dei sistemi presenti sul Web, individuando 3 diverse categorie:

- ◆ ***Sistemi a ipertesto statici***

Un sistema a ipertesto statico è caratterizzato dalla presenza di pagine standard, che hanno dei link e delle pagine statiche (fino a quando non vengono aggiornate, le pagine vengono visualizzate sempre e da tutti gli utenti allo stesso modo, indipendentemente da qualsiasi fattore).

- ◆ ***Applicazioni centrate su una base di dati***

Un’applicazione centrata su una base di dati è caratterizzata dalla creazione di pagine dinamiche, ma presenta ancora una struttura statica di link.

- ◆ ***Applicazioni Web dinamiche***

Un’applicazione Web dinamica fornisce invece sia la creazione dinamica delle pagine, sia una struttura dinamica dei link. È questa la categoria a cui appartengono i WIS.

Vantaggi derivanti dall’uso dei WIS

I benefici che derivano dall’uso dei WIS sono:

- ◆ ***Interoperabilità in ambienti eterogenei***

Il Web consente di promuovere l’interoperabilità di piattaforme, linguaggi e sistemi diversi: le applicazioni web possono essere eseguite su piattaforme software e hardware diverse.

- ◆ ***Servizi di business attraverso il Web***

I servizi Web possono essere usati per rendere accessibili a tutti i propri servizi di Business (ad esempio, l’azienda può rendere disponibili a venditori e clienti i propri cataloghi).

- ◆ ***Libertà di scelta***

Esiste un’ampia varietà di standard, che rende possibile alle singole organizzazioni di scegliere le configurazioni che meglio rispondono ai propri requisiti, senza richiedere agli sviluppatori di realizzare delle soluzioni proprietarie.

- ◆ ***Riduzione dei costi operativi***

I WIS permettono di estendere e riusare le funzionalità già esistenti, e così consentono la riduzione dei costi operativi.

- ◆ ***Espansione delle attività e aumento dell’efficienza***

L’architettura di un WIS fornisce all’azienda gli strumenti per espandere le proprie attività, aumentare la propria efficienza dei processi aziendali e aumentare l’efficienza verso i clienti. La gestione del business viene infatti automatizzata e il cliente ha un maggior numero di opzioni e scelte, oltre che una maggiore flessibilità.

2. Architettura dei WIS

Elementi essenziali di un WIS

Gli elementi essenziali di un WIS sono:

- ◆ ***Il Web server***

Il Web server si colloca tra l’utente finale e il sistema informativo aziendale. In particolare, è il componente che si occupa di gestire le richieste HTTP provenienti da Internet o dalla Intranet, restituendo al richiedente delle pagine statiche, oppure (mediante l’interazione con lo Script engine), il risultato dinamico di un’elaborazione.

- ◆ ***Lo Script engine***

Lo Script engine è un processo che esegue Script per generare delle pagine Web dinamiche. Tali pagine vengono poi inviate all’utente finale da parte del Web server.

- ◆ ***L’Application server***

È il middle tier, che implementa la logica di business dell’applicazione Web. Lo Script engine, per poter creare le pagine dinamiche, deve interagire con l’Application server.

- ◆ ***Il DBMS server***

Si occupa della gestione della base di dati. Lo Script engine deve interagire anche con il DBMS per poter generare le pagine dinamiche richieste dall’utente finale.

- ◆ ***I componenti di sicurezza***

Si tratta del firewall e di IDS (Intrusion Detection System):

- *Il firewall*

È un insieme di componenti hardware e software con l'obiettivo di collegare una rete ritenuta sicura con una che è invece ritenuta insicura. A tale scopo, il firewall controlla il traffico uscente e entrante in rete. Il firewall può essere di vari tipi:

- 1) *Packet filtering*

Il firewall sceglie se inoltrare o no un pacchetto in base a regole basate solo sull'header del pacchetto stesso. Questi firewall offrono ottime prestazioni, ma risulta difficile configurare le policy di sicurezza che essi richiedono.

- 2) *Application proxy*

L'application proxy è un processo eseguito da un host che connette due reti; i client della rete "interna" quindi non comunicano mai direttamente con host esterni alla rete, ma comunicano con il proxy, il quale richiede per conto loro i servizi all'host esterno alla rete.

Si hanno così dei filtri più ad alto livello, con la possibilità di impedire l'accesso a specifiche funzionalità o di introdurre meccanismi di autenticazione. Lo svantaggio è rappresentato dalla maggiore complessità dei calcoli che risultano necessari.

- 3) *Stateful inspection packet filter*

In questo caso il firewall sceglie se inoltrare un pacchetto oppure no in base non solo all'intestazione del pacchetto stesso, ma anche in base ai pacchetti precedentemente ricevuti e allo stato globale della connessione tra il client ed il server. È una soluzione intermedia tra le precedenti.

- *IDS (Intrusion Detection System)*

Un IDS è un sistema software o hardware che automatizza il processo di monitoraggio degli eventi di un sistema, al fine di individuare le intrusioni (tentativi di compromettere confidenzialità, integrità o disponibilità del sistema stesso).

Esistono due tipi di IDS:

- 1) *IDS network based*

Il monitoraggio avviene usando un insieme di sensori distribuiti nella rete, che analizzano i pacchetti che transitano nella rete stessa (ovviamente in modo nascosto).

Il vantaggio di questi IDS è dato dal fatto che con pochi sensori si può monitorare una rete molto grande, con un basso impatto sulla rete preesistente. Lo svantaggio è dato dalla scarsa efficienza, dall'impossibilità di analizzare dati cifrati e di rilevare se gli attacchi hanno successo.

- 2) *IDS host based*

In questo caso, su ogni singola macchina è installato l'IDS, in modo da raccogliere informazioni da tutti gli host.

Questi sistemi sono complessi da gestire e poco scalabili rispetto agli IDS network based, però consentono una maggior precisione nella rilevazione degli attacchi, con la possibilità di individuare gli host effettivamente colpiti; inoltre, possono operare anche con dati cifrati e sono in grado di stabilire l'esito degli attacchi.

Per combinare i vantaggi delle due tipologie di IDS, solitamente si installano gli IDS host based sui server con dati sensibili o che eseguono applicazioni critiche, mentre nei punti di ingresso e di uscita della rete viene installato un IDS network based.

I livelli logici e fisici di un WIS

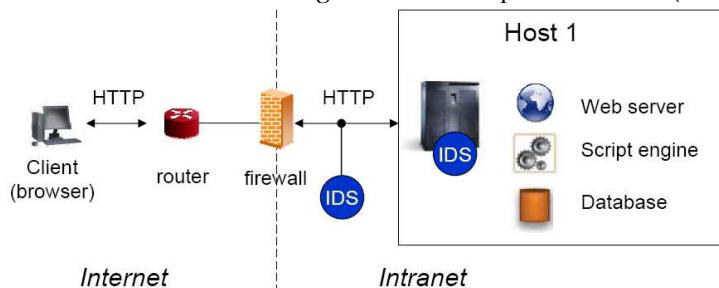
In un WIS, così come abbiamo visto per ogni altro sistema informativo, c'è la necessità di separare tra loro i vari livelli logici, detti anche *layer* o *layer applicativi*. Avremo perciò i 3 layer che caratterizzano ogni sistema informativo: presentazione, logica applicativa o di business, accesso ai dati. Si noti che al livello di presentazione non appartiene solo il browser, ma anche il server web e lo Script engine.

Da un punto di vista fisico, i 3 livelli logici possono essere allocati su uno stesso tier (architettura single-tier), oppure su più tier. In questa seconda ipotesi, abbiamo già brevemente descritto le architetture two-tier e three-tier. Abbiamo inoltre già accennato alla possibilità di usare architetture con un numero di tier n superiore di 3. Vogliamo ora concentrarci sulle architetture che realmente vengono utilizzate da un WIS.

Configurazione 2 tier single host

- ♦ *Descrizione della configurazione*

La configurazione 2 tier single host prevede che si abbia un'unica macchina fisica che supporta l'esecuzione di Web server, script, engine e DBMS. Tale macchina rappresenta perciò uno dei due tier, mentre l'altro è rappresentato dal client, che esegue il livello di presentazione (browser).



- ♦ *Vantaggi*

1. Semplicità di installazione e manutenzione, perché si ha una sola macchina da gestire.
2. Facilità di memorizzare dello stato dell'applicazione, all'interno dello Script engine o del database.

- ♦ *Svantaggi*

1. Scarsa affidabilità (se si blocca un componente, l'intero sistema non funziona).
2. Scarsa sicurezza (se un intruso supera il firewall, ha accesso all'intera macchina).
3. Costi elevati (per il principio del downsizing).
4. Prestazioni basse (per generare una pagina dinamica, il web server invoca uno script engine esterno che viene riavviato ad ogni invocazione, con un overhead computazionale evidentemente alto).

Configurazione 3 tier dual host

- ♦ *Descrizione della configurazione*

La configurazione 3 tier dual host può essere rappresentata come mostrato in figura:



In questa configurazione, il Web server e lo script engine sono ospitati su una stessa macchina, mentre il database è eseguito su una macchina dedicata. Il terzo tier è ancora una volta il client.

Per aumentare la sicurezza, viene inoltre installato un secondo firewall, a protezione del DBMS. Solitamente, il firewall esterno è un packet filtering, mentre quello interno è di tipo application proxy.

La Demilitarized Zone è un segmento di rete che si trova tra un segmento protetto e uno non protetto.

- ♦ *Vantaggi*

1. Si ha una maggior sicurezza (il DB ha una doppia protezione, grazie ai 2 firewall).
2. Una maggior scalabilità (si può intervenire separatamente sui 2 tier e il dimensionamento dei 2 tier risulta meno critico).

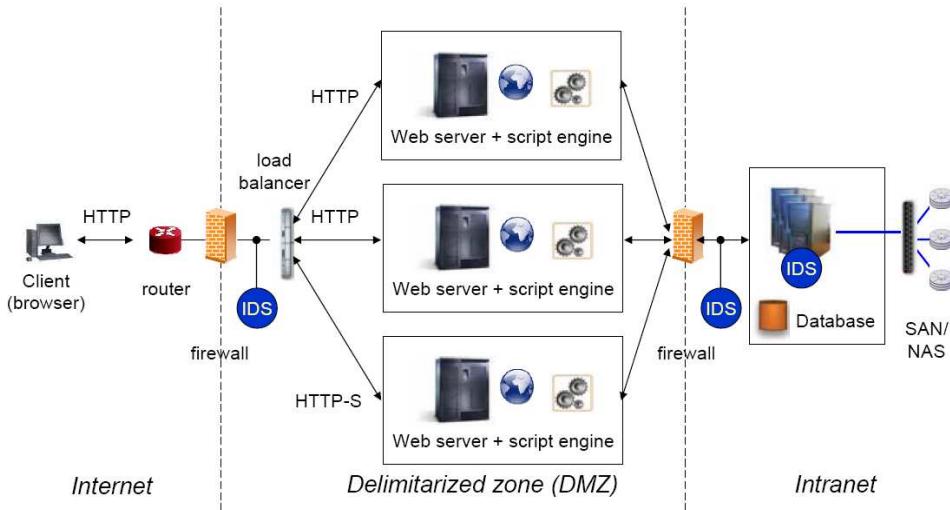
- ♦ *Svantaggi*

1. L'affidabilità rimane scarsa, perché il guasto di un solo componente comporta il non funzionamento dell'intero sistema.

Configurazione 3 tier con server farm

- ♦ *Descrizione della configurazione*

La configurazione 3 tier con server farm è descritta in figura:



Questa configurazione prevede che si abbiano ancora 3 tier, come nella precedente; tuttavia, il tier intermedio (oppure quello dedicato al database) non è occupato da un host, bensì da una server farm, all'interno della quale si usa solitamente una configurazione RACS o RAPS. In particolare:

1. Se si ha una server farm nel tier intermedio, si usa spesso la configurazione RACS shared-nothing.
2. Se la server farm è nel tier dedicato al DBMS, si usa spesso una configurazione shared-disk.

- ♦ *Vantaggi*

1. Aumento della disponibilità (se cade uno dei server, gli altri continuano a fornire il servizio, o al limite si ha un degrado parziale).
2. Aumenta la scalabilità.
3. Migliorano le prestazioni (a patto che il caro di lavoro sia efficacemente distribuito).

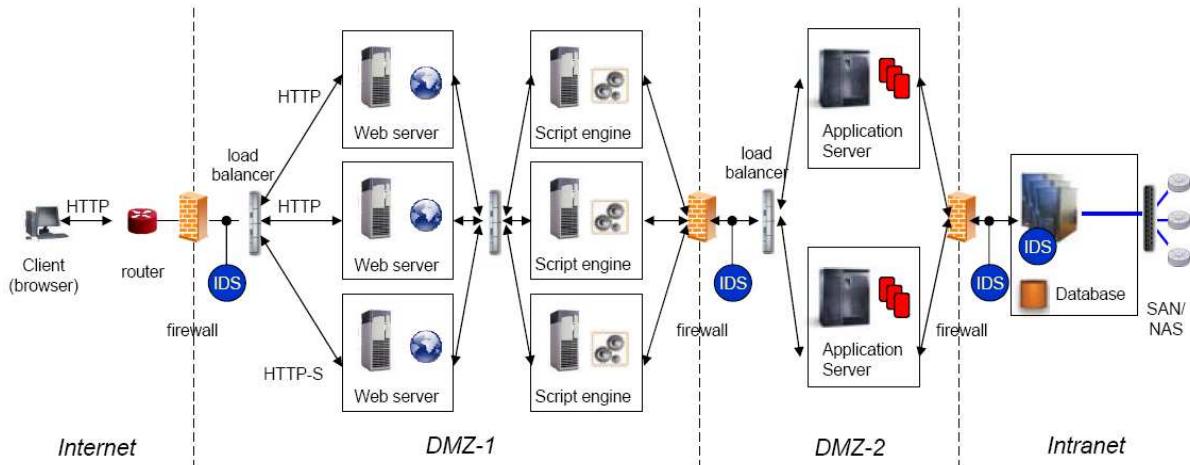
- ♦ *Svantaggi*

1. Si ha una elevata complessità di gestione e configurazione, oltre che di gestione del load-balancing (come meglio esplicitato nei 2 punti seguenti).
2. Se l'applicazione richiede la memorizzazione dello stato (è *stateful*), si ha una maggiore complessità, perché il load-balancer deve inoltrare tutte le richieste dello stesso client allo stesso server; di conseguenza, occorre analizzare al livello 7 del modello OSI ogni singolo pacchetto (perché l'IP non identifica univocamente un utente). In caso di un server però l'utente che stava colloquiando con quel server perderà lo stato: per evitarlo è possibile memorizzare lo stato nel DB, con conseguente aumento del carico di lavoro.
3. Se si usano protocolli sicuri (HTTPS o SSL), vengono generate delle chiavi di cifratura che sono note solo al client e al server che colloquiano tra loro, perciò non è possibile inoltrare a server diversi le richieste che provengono da uno stesso client.

Configurazione 5 tier con server farm

- ♦ *Descrizione della configurazione*

La configurazione 5 tier con server farm è rappresentata in figura:



Questa architettura è quella usata nei moderni centri di elaborazione dati. Si ha un tier per la base di dati, uno per l'application server, uno per lo Script engine, uno per il Web server e, infine, uno per la presentazione (il client, sul quale risiede il browser). All'interno dell'application server si ha un insieme di oggetti riusabili, che non vengono invocati solo dal Web, ma anche dalle altre applicazioni aziendali.

- ♦ *Vantaggi*

1. Maggiore sicurezza: si hanno infatti tre livelli di firewall e 2 diverse zone demilitarizzate (la seconda, più sicura, è quella in cui si trova l'application server, che richiede maggior protezione proprio perché contiene oggetti invocati da tutte le applicazioni aziendali).
2. Prestazioni migliori, grazie al load-balancing dinamico.
3. Massima scalabilità.
4. Elevata disponibilità.

- ♦ *Svantaggi*

1. Elevata complessità di gestione e manutenzione;
2. Costi elevati.

Architettura del client del WIS

Per quanto riguarda il client (che abbiamo fino ad ora trascurato), possiamo fondamentalmente distinguere (come nelle altre tipologie di sistemi informativi) tra architetture thin client e fat client. Nel caso di un WIS, l'architettura thin client prevede che il browser si limiti a visualizzare la pagina HTML; in questo modo, si è completamente liberi dal tipo di browser che viene usato.

Se invece si sceglie di usare un'architettura fat, significa che all'interno della pagina HTML sono inserite anche delle porzioni applicative da far eseguire al browser (es.: Apple Java o controlli ActiveX). In questo modo, migliora l'usabilità del sistema, però si deve avere un maggior controllo sul tipo di browser che viene usato per accedere al sistema informativo.

Nel caso di architetture distribuite, è anche possibile utilizzare gli stub di oggetti distribuiti, che instaurano una connessione permanente con un oggetto lato server corrispondente.

3. Classificazione dei servizi elettronici in rete

Le possibili modalità di classificazione dei WIS e, in generale, dei servizi elettronici in rete, sono svariate. Vediamo adesso i principali criteri in base ai quali operare questa classificazione.

Classificazione dell'UE

L'Unione Europea ha classificato i servizi elettronici in rete, individuando (all'interno del *Libro Verde*) le categorie di seguito elencate e descritte:

- ◆ ***Servizi di informazione***

I servizi di informazione sono servizi il cui obiettivo è quello di fornire accesso ad informazioni strutturate e classificate. Il sistema è costituito da pagine statiche o dinamiche e può presentare inoltre link ad altre risorse informative. In ogni caso, l'utente non inserisce contenuto informativo nel sistema.

- ◆ ***Servizi di comunicazione***

I servizi di comunicazione sono servizi che favoriscono e supportano la comunicazione di gruppi di utenti. Essi sono stati implementati attraverso vari tipi di applicazioni come e-mail, chat, ma anche applicazioni Web come newsgroup, blog, forum,

Per certi versi, questi servizi potrebbero essere assimilati ai servizi informativi, ma in questo caso le informazioni che vengono mostrate riguardano la comunità, ed è la comunità stessa ad inserirle.

In questi servizi, l'utente è identificato (eventualmente solo con un'identità fittizia) e le comunicazioni dell'utente possono rimanere memorizzate nel sistema informativo.

- ◆ ***Servizi transazionali***

I servizi transazionali sono servizi che supportano gli utenti nell'esecuzione di transazioni, come l'acquisto di beni o servizi. L'utente, che viene solitamente identificato dal sistema, interagisce con i DBMS e con i servizi applicativi.

La principale criticità di questa tipologia di servizi è costituita dalle problematiche di sicurezza e riservatezza dei dati trasmessi e memorizzati.

Classificazione UE in base ai livelli di interazione

L'Unione Europea ha inoltre classificato i sistemi elettronici in rete in base ai vari livelli di interazione:

- ◆ ***Livello 1***

In questo caso, sono disponibili on-line solo le informazioni necessarie per avviare la procedura che porta all'erogazione del servizio. L'utente trova informazioni sull'organizzazione, sulle attività svolte ed i contatti per richiedere ulteriori informazioni via e-mail, telefono o posta.

- ◆ ***Livello 2***

Nei servizi di livello 2, è possibile scaricare e stampare on-line i moduli necessari ad avviare la procedura che porta all'erogazione del servizio. Non è possibile però inviare on-line il modulo compilato; l'utente è quindi obbligato a recarsi fisicamente presso gli uffici dell'organizzazione, o ad usare un vecchio canale di comunicazione (ad esempio fax o posta) per la consegna della richiesta.

- ◆ ***Livello 3***

Questo tipo di servizio prevede l'interazione in due sensi (per esempio, si ha la possibilità di avviare on-line la procedura, con compilazione di moduli elettronici ed autenticazione utente). Ciò rappresenta una prima forma di fruizione remota del servizio, che svincola l'utente dai vecchi canali di comunicazione per la consegna della richiesta.

- ◆ ***Livello 4***

In questo caso, è prevista l'esecuzione on-line di un'intera procedura (per esempio, nel caso di una transazione di commercio elettronico, tutte le fasi che vanno dall'ordine della merce, al pagamento elettronico, alla consegna). Questo livello è caratterizzato dall'assenza di moduli cartacei per erogare il servizio e non necessita di spostamenti fisici da parte dell'utente, che può gestire l'intera procedura dal terminale d'accesso.

Si osserva quindi che nei livelli di interazione 1 e 2, l'utente non può inviare alcuna informazione al sistema informativo, mentre nei livelli di interazione 3 e 4 si ha un flusso di informazioni provenienti dall'utente.

Classificazione in base alle modalità d'accesso

Un'altra classificazione per i servizi in rete si basa sulle *modalità di accesso ai siti*. Avremo così:

- ◆ **Sito Internet**

Un sito Internet è accessibile da tutta la rete, e quindi gli utenti sono disparati e non noti a priori. Su questi siti l'obiettivo principale è *l'accessibilità* (garantire la possibilità di accedere al sistema all'insieme più ampio possibile di utenti).

Dovendo permettere l'accesso a browser diversi che vengono eseguiti potenzialmente su piattaforme diverse, dovranno essere adottate *solo* tecnologie standard e si dovrà assumere l'esistenza di una connessione a banda stretta verso l'utente finale.

- ◆ **Sito Intranet**

L'accesso è consentito solo all'interno dell'organizzazione (gli utenti sono perciò i dipendenti dell'organizzazione stessa). Perciò, si ha anche un'elevata sicurezza. Questi siti vengono usati per:

1. Gestire la conoscenza aziendale, distribuendo informazioni nell'azienda stessa
2. Consentire la collaborazione fra reparti e la partecipazione degli utenti a processi decisionali.

A questo livello, si ha un forte controllo sulla configurazione della piattaforma hardware della rete e dei client del sistema e possono essere adottate architetture client thin o fat a seconda dell'applicazione.

- ◆ **Sito Extranet**

Un sito Extranet è accessibile da un gruppo di utenti ben identificato, ad esempio clienti e fornitori. È lo strumento di scambio di informazioni concordate tra soggetti diversi tramite accesso ad archivi (ad esempio per rendere visibili i livelli delle scorte in magazzino) e di accesso a funzionalità predefinite che consentono l'integrazione della supply chain di organizzazioni diverse.

Dato il loro alto livello di standardizzazione, questo è uno scenario applicativo in cui l'adozione dei Web Service risulta molto promettente.

Una Extranet è un'estensione dell'Intranet aziendale che consente l'accesso a utenti esterni all'azienda. In sistemi Extranet, oppure per applicazioni Web che implementano un livello di interazione elevato (3 o 4), esistono problemi di sicurezza e privatezza delle informazioni; ciò impone l'introduzione di meccanismi di certificazione e autenticazione degli utenti e di crittografia per la trasmissione dei dati.

Intranet ed Extranet si basano sulla stessa tecnologia di Internet, adottano cioè principalmente i protocolli TCP/IP per la comunicazione ed HTTP per l'accesso alle applicazioni.

| | Sito Internet | Sito Intranet | Sito Extranet |
|-----------------------|-------------------------------------------------|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Utenti | Noti con approssimazione | Dipendenti dell'organizzazione | Dipendenti di più organizzazioni |
| Compiti | Informativo, servizi ai cittadini | Supporto al lavoro dei dipendenti, gestione dei processi | Integrazione di processi delle diverse organizzazioni |
| Larghezza banda | bassa | Alta, affidabile | Alta, affidabile |
| Compatibilità | Diversi browser, massima accessibilità del sito | Scelta a priori del browser | Scelta a priori della tecnologia della propria organizzazione e conoscenza di quella adottata dai partner |
| Quantità informazioni | Poche e in tempi rapidissimi | Quantità elevate, per gestione processi operativi | Quantità elevate, integrazione della supply chain |

Classificazione in base agli utenti del servizio

Esiste poi una classificazione basata sugli obiettivi e sulle tipologie di utenti a cui è rivolta un'applicazione Web. Si parla di:

- ◆ **Applicazioni B2B (Business to Business) oppure G2G (Government to Government)**

Sono applicazioni rivolte principalmente alla cooperazione tra imprese (nel caso B2B), o Pubbliche Amministrazioni (nel caso G2G). Queste applicazioni ricadono pertanto nel caso delle Extranet.

- ◆ **Applicazioni B2C (Business to Consumer) o G2C (Government to Citizen)**

Sono applicazioni rivolte al singolo cliente o cittadino (es.: applicazioni per il commercio elettronico).

Capitolo 9: Tecnologie Web

1. Introduzione alle tecnologie Web

Introduzione

Giunti a questo punto della trattazione, è opportuno introdurre le principali tecnologie che si possono utilizzare per la realizzazione di un WIS, ovvero gli strumenti che si hanno a disposizione per la realizzazione concreta del WIS stesso.

Per prima cosa, possiamo operare una distinzione tra pagine statiche e dinamiche:

- ♦ *Pagine statiche*

Una pagina statica è una pagina che viene presentata sempre esattamente allo stesso modo, fino quando la versione del corrispondente file memorizzato sul Web server non viene modificata, e indipendentemente dall'utente che l'ha richiesta.

- ♦ *Pagine dinamiche*

Una pagina dinamica viene generata nel momento in cui viene richiesta, molto spesso sulla base di dati che vengono inviati dall'utente al Web Server. La richiesta HTTP inviata dal Web browser contiene perciò, oltre all'URL desiderato, anche tali informazioni aggiuntive; quando il Web Server riceve la richiesta, esso esegue un'applicazione che produce come risultato la pagina HTML poi inviata al client dal Web Server.

Naturalmente, un WIS è costituito principalmente da pagine dinamiche.

Tecnologie per la realizzazione di pagine dinamiche

Le principali tecnologie che possono essere usate per la realizzazione di pagine dinamiche sono:

- ♦ *Tecnologia CGI*

La tecnologia CGI è stata la prima ad essere sviluppata (può essere considerata quindi la capostipite delle seguenti tecnologie). Essa prevede di avere una o più applicazioni scritte in un linguaggio qualsiasi, che generano sullo standard output una pagina HTML.

- ♦ *Linguaggi di scripting*

Esistono poi dei linguaggi che consentono di realizzare delle pagine HTML contenenti del codice eseguibile lato server: il server esegue tale codice e si ottiene così una pagina HTML che viene mandata al client. Appartengono a questa categoria PHP, ASP e JSP.

- ♦ *Servlet Java*

Un'altra soluzione è quella di utilizzare le cosiddette *servlet Java*; una Servlet è un componente Java definito da opportune specifiche, che può essere eseguito da un Web Server Java enabled per l'implementazione di nuovi servizi.

Oltre a tali tecnologie, possiamo aumentare la dinamicità della pagina inserendo del codice eseguibile dal lato del client. A tale scopo, le tecnologie più usate sono JavaScript e VBScript.

2. Realizzazione di pagine statiche: HTML

Introduzione ad HTML

Il linguaggio HTML (Hyper Text Markup Language) è un linguaggio di marcatura o di markup. Si tratta perciò di un linguaggio utilizzato per creare ipertesti: HTML è in grado di definire la struttura per la presentazione e il contenuto di pagine statiche, visualizzabili attraverso un Web browser. Tali pagine sono dette appunto pagine HTML.

I tag

Il linguaggio html utilizza una serie di marcatori, detti anche tags, che consentono di definire la struttura logica del documento, le relazioni che legano tra loro i diversi documenti, o semplicemente gli aspetti di presentazione del documento. Possiamo perciò affermare che una pagina HTML è una sequenza di tag che racchiudono porzioni di testo, e che ogni tag rappresenta un “comando” (non è una vera azione da eseguire, ma una specifica che riguarda il modo in cui il browser deve visualizzare il contenuto).

Formalmente, si possono avere:

- ♦ **Tag di apertura e di chiusura**

In questo caso, si ha dapprima un tag del tipo (tag di apertura):

<nome-tag>

E poi si ha un tag di chiusura:

</nome-tag>

Tra il tag di apertura e quello di chiusura potremo trovare del testo e/o una serie di altri tag.

Nel tag di apertura possiamo inoltre avere uno o più attributi, secondo il formato:

<nome-tag attributo1 = "valore1" attributo2 = "valore2">

- ♦ **Tag vuoti (empty tag)**

Un empty tag è un tag di apertura che non prevede che ci sia un corrispondente tag di chiusura (di conseguenza, non potremo avere testo o altri tag al suo interno). Essendo un tag di apertura, può avere degli attributi. Un empty tag è identificato dal fatto che termina con il simbolo /. La struttura è quindi:

<nome-tag />

Nel caso in cui sia previsto, ad esempio, un attributo, avremo:

<nome-tag attributo = "valore">

Struttura di una pagina HTML

- ♦ *Struttura di base*

La struttura di ogni pagina HTML è riassumibile mediante il seguente schema:

```
<html>
  <head>
    <title>
      Titolo visualizzato sulla barra del titolo del browser.
    </title>
    <meta>
      Metadati
    </meta>
  </head>
  <body>
    Testo e tag che costituiscono il corpo del documento html.
  </body>
</html>
```

- ♦ *L'intestazione*

L'intestazione (racchiusa tra i tag `<head>` e `</head>`) è in realtà opzionale, ma è fortemente consigliato inserirla (tutte le restanti parti sono invece obbligatorie). All'interno dell'intestazione abbiamo:

1. Il titolo visualizzato nella barra del titolo del browser, racchiuso tra i tag `<title>` e `</title>`.

2. I metadati, che definiscono il contenuto della pagina web. In particolare, potremo avere:

- a) Il tipo di codifica. Ad esempio, se scriviamo

```
<meta http-equiv = "Content-Type" content = "text/html; charset=iso-8859-1">
```

Stiamo indicando che nella pagina avremo un testo contenente caratteri accentati italiani.

- b) Coppie del tipo (nome, valore), del tipo:

```
<meta name = "nome" content = "valore">
```

Il nome può essere, tra l'altro, *keywords* (parole chiave usate dai motori di ricerca per catalogare le pagine web), *author* (autore della pagina web) oppure *description* (descrizione della pagina).

- ♦ *Attributi del tag body*

Colore di sfondo

Tramite costante numerica: `<BODY BGCOLOR = "#RRGGBB">`

Tramite nome del colore: `<BODY BGCOLOR = "ColorName">`

Dove i colori sono riassunti nella tabella seguente:

| Colore | Formato #RRGGBB | Formato ColorName |
|--------|-----------------|-------------------|
| Bianco | #FFFFFF | white |
| Nero | #000000 | black |
| Rosso | #FF0000 | red |
| Blu | #0000FF | blue |
| Verde | #00FF00 | green |

Immagine di sfondo Sfondo della pagina: `<BODY BACKGROUND="percorso/nome.est">`

N.b.: il percorso può essere assoluto o, come è preferibile, relativo.

Colore dei link

Link generici:

```
<BODY LINK = "#RRGGBB">
```

Link attivi:

```
<BODY ALINK = "#RRGGBB">
```

Link visitati:

```
<BODY VLINK = "#RRGGBB">
```

Colore del testo

Testo di tutta la pagina:

```
<BODY TEXT = "#RRGGBB">
```

Tags di formattazione del testo e inserimento immagini

- ♦ *Paragrafi e intestazioni*

Il tag P

Identifica del testo normale. <P> Nuovo paragrafo </P>

L'attributo align consente di impostare l'allineamento del testo (di default a sinistra):

```
<P ALIGN = "RIGHT"> Testo a destra </P>
<P ALIGN = "LEFT"> Testo a sinistra </P>
<P ALIGN = "CENTER"> Testo centrato </P>
```

Due paragrafi successivi sono distanziati da un'interlinea doppia.

A capo semplice

Si va a capo restando nello stesso paragrafo. Riga1
 Riga 2

Intestazioni

Rappresentano i sei livelli con i quali è possibile strutturare un documento HTML.

Anche in questo caso, si può avere l'attributo align.

Questo testo è racchiuso tra i tag H1

Questo testo è racchiuso tra i tag H2

Questo testo è racchiuso tra i tag H3

Questo testo è racchiuso tra i tag H4

Questo testo è racchiuso tra i tag H5

Questo testo è racchiuso tra i tag H6

Questo testo è racchiuso tra i tag P

- ♦ *Tags di modifica del carattere*

Testo in grassetto

 Testo da visualizzare in grassetto

Testo in corsivo

<I> Testo da visualizzare in corsivo </I>

Testo sottolineato

<U> Testo da visualizzare sottolineato </U>

Colore del testo

Tramite costante numerica: Testo

Tramite nome del colore: Testo

Dimensioni del testo

 Testo

N.b.: la dimensione n del carattere deve essere un numero intero tra 1 e 7.

Scelta del font

 Testo

N.b.: il secondo nome di font (possono essercene anche più di due) è facoltativo e viene utilizzato nel caso in cui non sia presente il primo font (o tutti quelli precedenti, se non fosse il secondo) tra quelli installati sulla macchina dalla quale si cerca di visualizzare la pagina.

Spaziatura fissa

<TT> Testo con spaziatura fissa </TT>

- ♦ *Tag per l'inserimento di immagini*

Tag IMG

Alcuni formati di immagini non sono supportati (ad esempio, BMP e TIFF).

Attributo bordo

Attributi dimensione

Aampiezza e larghezza possono essere espresse in due modi:

1. In percentuale, rispetto all'intero monitor. Es.: WIDTH = "40%"
2. Tramite il numero di pixel. Es.: WIDTH = "200"

Tags per l'inserimento di elenchi e liste

Elenchi puntati

```
<UL TYPE = "Tipo_Elenco_Puntato">
    <LI> Primo punto </LI>
    <LI> Secondo punto </LI>
    [...]
</UL>
```

L'attributo TYPE è facoltativo. Se viene omesso, viene usato come simbolo il pallino. Gli altri valori sono:

| Simbolo | attributo TYPE |
|-------------|----------------|
| • Elemento. | TYPE = DISC |
| ▪ Elemento. | TYPE = SQUARE |
| ◦ Elemento. | TYPE = CIRCLE |

Elenchi numerati

```
<OL TYPE = "Tipo_Elenco_Numerato">
    <LI> Primo punto </LI>
    <LI> Secondo punto </LI>
    [...]
</OL>
```

L'attributo TYPE è facoltativo. Se viene omesso, viene visualizzato un elenco numerato dall'1. Gli altri possibili valori sono:

| Simbolo | attributo TYPE |
|---------------|----------------|
| A. Elemento. | TYPE = A |
| B. Elemento. | |
| a. Elemento. | TYPE = a |
| b. Elemento. | |
| I. Elemento. | TYPE = I |
| II. Elemento. | |
| 1. Elemento. | TYPE = i |
| 2. Elemento. | |
| 1. Elemento. | TYPE = 1 |
| 2. Elemento. | |

Definition List

```
<DL>
    <DT> Termine da definire (visualizzato come titolotto) </DT>
    <DD> Definizione (visualizzata con rientro) </DD>
    [...]
</DL>
```

Tags per l'inserimento di link

Un link è definito dai due elementi che collega: la sorgente e la destinazione del link. La sorgente è una porzione di testo o un'immagine, ed è identificata dal fatto che è racchiusa tra i tag `<A>` e ``; se si clicca sulla sorgente, si arriva alla destinazione, che può essere un'immagine, un'altra pagina, in generale un'altra risorsa, oppure anche un punto all'interno della stessa pagina in cui si trova la sorgente (ancora).

Link semplice ad un altro oggetto

```
<A HREF = "indirizzo URL"> Sorgente </A>
```

Invio di e-mail

```
<A HREF="mailto:indirizzo"> Sorgente </A>
```

Link a punto marcato

Il punto marcato (ancora) è identificato sempre usando il tag A:

```
<A NAME = "nome_punto">
```

Per poi definire il link verso l'ancora nella stessa pagina, scriveremo:

```
<A HREF = "#nome_punto">
```

Oppure, da un'altra pagina:

```
<A HREF = "indirizzo_pagina#nome_punto"> Sorgente </A>
```

Tags per l'inserimento di tabelle

Di seguito sono riportati i tag relativi alle tabelle:

Apertura e chiusura tabella

```
<TABLE ALIGN = LEFT|RIGHT|CENTER VALIGN = TOP|BOTTOM|MIDDLE HEIGHT = alt  
WIDTH = amp CELLPADDING = n CELLSPACING = n BGCOLOR = colore>  
  
</TABLE>
```

N.b.: ALIGN = allineamento orizzontale

VALIGN = allineamento verticale

HEIGHT = altezza tabella (in numero di pixel o in percentuale)

WIDTH = larghezza tabella (in numero di pixel o in percentuale)

CELLPADDING = spazio tra il bordo della cella e il suo contenuto

CELLSPACING = spazio tra due celle adiacenti

BGCOLOR = colore di sfondo della tabella (in formato "#RRGGBB" o tramite nome colore)

Apertura e chiusura riga (all'interno della tabella)

```
<TR ALIGN = LEFT|RIGHT|CENTER VALIGN = TOP|BOTTOM|MIDDLE HEIGHT = alt  
WIDTH = amp BGCOLOR = colore>  
  
</TR>
```

Per definire una cella con uno stile diverso, anziché il tag `<TD>` si può usare il tag `<TH>` (solitamente lo si fa per le celle che costituiscono la riga di intestazione).

Apertura e chiusura cella (all'interno della riga)

```
<TD ALIGN = LEFT|RIGHT|CENTER VALIGN = TOP|BOTTOM|MIDDLE HEIGHT = alt  
WIDTH = amp BGCOLOR = colore COLSPAN = n ROWSPAN = n>  
  
</TD>
```

N.b.: COLSPAN = nel caso in cui si vogliano unire più celle, rappresenta il numero di celle che vogliono essere unite in larghezza (il numero di colonne coinvolte).

ROWSPAN = nel caso in cui si vogliano unire più celle, rappresenta il numero di celle che vogliono essere unite in altezza (il numero di righe coinvolte).

- ♦ **Esempio di tabella**

Di seguito è riportato un semplice esempio di tabella:

```
<TABLE>  
  <TR>  
    <TD>  
      Prima riga, prima colonna  
    </TD>  
    <TD>  
      Prima riga, seconda colonna  
    </TD>  
  </TR>  
  <TR>  
    <TD>  
      Seconda riga, prima colonna  
    </TD>  
    <TD>  
      Seconda riga, seconda colonna  
    </TD>  
  </TR>  
</TABLE>
```

Tags per l'inserimento di form

Le forms sono dei moduli che consentono all'utente l'inserimento di dati, in modo da fornire l'input ad un'applicazione.

- ♦ *Tag di apertura e chiusura di una form*

```
<FORM ACTION = "nome applicazione" METHOD = "GET|POST" NAME = "nome form">
</FORM>
```

N.b.: ACTION = Consente di specificare il nome (se necessario comprensivo del percorso, assoluto o relativo) dell'applicazione a cui inviare i dati inseriti nella form.

METHOD = Attributo che consente di specificare la modalità di trasferimento dei dati:

GET = i dati sono inviati aggiungendoli all'indirizzo visualizzato sull'apposita barra del browser, nel formato:

url?nome_parametro1=valore1&nome_parametro2=valore2

POST = i dati sono inviati memorizzandoli nel body della richiesta HTTP.

Usando il metodo GET si ha un limite massimo nella dimensione dei dati (1024 byte), i quali inoltre sono facilmente visibili e devono necessariamente essere di tipo testo. Tali problemi non si hanno usando il metodo get.

- ♦ *Tags per l'inserimento dei campi*

TextBox

```
<INPUT TYPE = "text" NAME = "nome_var_di_destinazione" ID = "identificativo_campo"
      SIZE = "lunghezza_della_casella" MAXLENGTH = "num_max_caratteri_inseribili"
      VALUE = "testo_predefinito_nella_casella"/>
```

TextBox per Password (visualizza asterischi al posto del testo inserito)

```
<INPUT TYPE = "password" NAME = "nome_var_di_destinazione" ID = "id_campo"
      SIZE = "lunghezza_della_casella" MAXLENGTH = "num_max_caratteri_inseribili"
      VALUE = "testo_predefinito_nella_casella"/>
```

Radio

```
<INPUT TYPE = "radio" NAME = "variabile" ID = "id_campo"
      VALUE = "valore_se_selezionata_op1" CHECKED/> Opzione 1
<INPUT TYPE = "radio" NAME = "variabile" ID = "id_campo"
      VALUE = "valore_se_selezionata_op2"/> Opzione 2
[...]
```

N.b.: CHECKED = attributo facoltativo che indica che quella radio è selezionata di default.

Può essere presente su una sola radio.

Checkbox

```
<INPUT TYPE = "checkbox" NAME = "variabile1" VALUE = "valore_se_selezionata"
      ID = "id_campo" CHECKED /> Opzione 1
<INPUT TYPE = "checkbox" NAME = "variabile2" VALUE = "valore_se_selezionata"
      ID = "id_campo" CHECKED /> Opzione 2
[...]
```

N.b.: CHECKED = attributo facoltativo che indica che quella radio è selezionata di default.

Può essere presente su più checkbox della lista.

Tendina

```
<SELECT NAME = "variabile" ID = "id_campo">
  <OPTION VALUE = "valore_se_scelta_op1"> Opzione 1 </OPTION>
  <OPTION VALUE = "valore_se_scelta_op2"> Opzione 2 </OPTION>
[...]
</SELECT>
```

TextArea

```
<TEXTAREA NAME = "variabile" COLS = "n_caratteri_per_riga"
          ROWS = "n_caratteri_per_colonna" ID = "id_campo"> Testo predefinito </TEXTAREA>
```

Campo nascosto

```
<INPUT TYPE = "hidden" NAME = "variabile" VALUE = "valore" ID = "id_campo"/>
```

Pulsante di submit (Ok)

```
<INPUT TYPE = "submit" VALUE = "testo da visualizzare" NAME = "nome"/>
```

Pulsante di reset (Annulla)

```
<INPUT TYPE = "reset" VALUE = "testo da visualizzare" NAME = "nome"/>
```

3. JavaScript

Che cos'è JavaScript?

Le pagine HTML possono essere arricchite con dei semplici script. Uno dei linguaggi disponibili a tale scopo è JavaScript. Il codice JavaScript viene inglobato all'interno di una pagina HTML, mediante il tag script, inserito nell'intestazione (solitamente dopo il title).

```
<HTML>
  <HEAD>
    <TITLE> Titolo della finestra </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
      Codice javascript
    </SCRIPT>
  </HEAD>
  <BODY> Corpo della pagina </BODY>
</HTML>
```

Uno script viene eseguito a lato client (cioè dal browser) quando è richiamato da un altro script, oppure viene intercettato l'evento al quale lo script è associato.

Come si scrive il codice JavaScript?

Per scrivere del codice JavaScript, si costruiscono delle semplici funzioni, del tipo:

```
function NomeFunzione(){
  //istruzioni
}
```

Si usa il punto e virgola come terminatore di ogni istruzione e si possono inserire commenti mediante il doppio slash (//). Le variabili non sono tipizzate, ma vengono dichiarate scrivendo:

```
var nome_variabile = valore_di_default;
```

Naturalmente, il valore di default può essere omesso.

Si ha a disposizione un oggetto document, che rappresenta il contenuto della pagina HTML. Un riferimento a tale oggetto viene ottenuto mediante la proprietà document dell'oggetto window, che è globalmente disponibile in ogni funzione JavaScript.

Attraverso la notazione puntata, si può poi accedere alla form, e poi ancora ai singoli campi contenuti al suo interno e agli attributi di tali oggetti. Inoltre, l'oggetto document ha il metodo getElementById(), che consente di ottenere l'elemento associato ad un certo ID.

Eventi predefiniti

Gli eventi predefiniti in JavaScript sono:

- onAbort si attiva all'interruzione del caricamento di un'immagine.
- onClick si attiva alla pressione del mouse su quell'oggetto.
- onChange si attiva quando cambia il valore di un elemento.
- onError si attiva quando si verifica un errore.
- onLoad si attiva al caricamento dell'oggetto
- onMouseOver si attiva quando con il mouse si passa sopra ad un elemento.
- onReset si attiva quando si fa il reset.
- onSelect si attiva quando si seleziona un elemento di input.
- onSubmit si attiva quando si fa il submit.
- onUnload si attiva quando si chiude la pagina.

Si può associare una funzione ad un evento relativo ad un oggetto aggiungendo un attributo con il nome dell'evento all'interno del tag che definisce l'elemento in questione; il valore attribuito all'attributo è il nome della funzione JavaScript attivata di conseguenza.

Esempio

Un banale esempio di utilizzo di JavaScript è di seguito riportato:

```
<html>
  <head>
    <title>
      Semplice esempio di uso di JavaScript.
    </title>
    <script language = "JavaScript">
      function VisualizzaTesto() {
        window.document.getElementById("testo1").innerHTML =
          window.document.prova.testo.value;
      }
    </script>
  </head>
  <body>
    <FORM name = "prova">
      <TABLE>
        <TR>
          <TD>
            <input type="text" name="testo" value="a" OnChange='Prova()' />
          </TD>
          <TD id="testo1">a</TD>
        </TR>
      </TABLE>
    </FORM>
  </body>
</html>
```

4. Le sessioni utente

Che cos'è una sessione?

Una sessione rappresenta un singolo uso coerente di un sistema Web e richiede solitamente l'accesso a diverse pagine e una forte interazione con la logica applicativa del sistema. Il protocollo HTTP è stateless, e ciò rende più complessa la realizzazione delle sessioni.

I cookie

Per risolvere il problema, il W3C ha introdotto un meccanismo per gestire lo stato nel client: i cookie. Un cookie è un insieme di dati memorizzato sul client, su richiesta del Web Server. I dati vengono forniti dal browser ad ogni richiesta HTTP. La dimensione massima di un cookie è di 4 Kbyte e all'interno dei cookie sono memorizzati i seguenti dati:

1. Il nome del cookie;
2. Il valore del cookie;
3. La scadenza (l'intervallo di tempo in secondi per il quale il browser deve mantenere il cookie);
4. Il dominio di appartenenza del cookie;
5. Il percorso, espresso come path nel dominio;
6. La richiesta di connessione sicura.

Un server richiede al browser di memorizzare i cookie aggiungendo all'intestazione HTTP delle linee del tipo:

```
Set-Cookie: sessionID=12345; path=/; expires 8/6/2005
Set-Cookie: totaleCarrello=0; path=/; expires 8/6/2005
```

Se il browser è configurato per accettare i cookie, queste stringhe vengono accettate e memorizzate sul client in una posizione dipendente dallo specifico browser. Alla successiva richiesta il browser invierà nella richiesta HTTP la seguente stringa:

```
Cookie: sessionID=12345; totaleCarrello=0
```

Se il dominio non è impostato esplicitamente, viene considerato come dominio il nome di dominio completo del documento che ha creato il cookie. Il cookie viene trasmesso solo quando il browser richiede l'accesso ad una risorsa appartenente al dominio ed al percorso specificato nel cookie. In questo modo un server del dominio www.miodominio.com non potrà impostare un cookie valido per www.altrodominio.com.

Possibili modalità per la gestione delle sessioni utente

Le modalità possibili per gestire le sessioni utente sono 4:

- ◆ ***Inserimento di tutte le variabili di stato nel cookie***

In questo modo, ad ogni richiesta il client deve inviare, tramite cookie, tutti i dati dello stato della sessione corrente. Questa soluzione comporta alcuni problemi:

- a) La dimensione del cookie è limitata a 4 Kbyte.
- b) Si può avere un numero massimo di cookie pari a 20.
- c) I dati possono essere solo in formato di testo, e non possono essere dati strutturati.
- d) Alcuni browser impediscono l'uso dei cookie.

Il vantaggio è invece dato dalla grande semplicità di questa soluzione.

- ◆ ***Inserimento di una chiave univoca memorizzata come cookie***

Tale soluzione prevede che le informazioni vengono memorizzate lato server, e che il client memorizzi su un cookie solamente un identificativo numerico: in tal modo, il client invierà come cookie solamente l'identificativo, e il server sarà in grado di identificare univocamente l'utente da cui proviene la risposta. Al termine della sessione, i dati sul server devono essere eliminati.

Si superano così i problemi della dimensione e del numero dei cookie, ma rimane il problema relativo alle impostazioni del browser.

- ◆ ***Passaggio di tutti i dati dello stato come parametri***

In alternativa, è possibile utilizzare dei parametri nascosti che passano le informazioni riguardanti lo stato da una pagina all'altra, mediante il metodo hidden oppure get (si usano cioè dei campi hidden). In particolare quindi questa soluzione consiste nel passare le variabili che riguardano l'intero stato mediante campi nascosti.

- ◆ ***Passaggio di una chiave univoca come parametro***

L'ultima soluzione consiste nel passare come dato nascosto solamente un identificativo univoco, memorizzando però tutti i restanti dati all'interno del server, e accedendo a tali dati mediante il valore di chiave ricevuto come parametro mediante i metodi get oppure post.

5. La tecnologia CGI

Come funziona?

Come abbiamo detto, la tecnologia capostipite per la realizzazione di contenuti dinamici sul Web è stata la tecnologia CGI (Common Gateway Interface). In questo caso, si suppone di avere sulla macchina server una sere di applicativi in grado di ricevere in ingresso alcuni dati e di restituire in uscita una pagina HTML dipendente dai dati ricevuti. Ogni applicazione è infatti vista come una risorsa a disposizione, e perciò possiede un URL.

Nel dettaglio, le applicazioni devono essere contenute in una opportuna cartella. Ad esempio, se si utilizza come Web Server l'applicazione Apache, la cartella è cgi-bin. L'URL sarà quindi del tipo:

`http://www.dominio.it/cgi-bin/esempio.py?a=prova`

Il reale percorso della cartella cgi-bin può comunque essere impostato mediante il file di configurazione del Web Server. Quando l'utente digita sul proprio browser questo indirizzo e invia la richiesta HTTP, la richiesta viene inviata al Web Server. Il Web server riconosce che deve eseguire l'applicazione esempio.py, che si trova all'interno della cartella cgi-bin; inoltre, il Web Server passa a tale applicazione il valore "prova" da attribuire al parametro a. L'applicazione restituisce un risultato sullo standard output: tale risultato viene reindirizzato dal Web Server, e inviato mediante un messaggio HTTP al client da cui era arrivata la richiesta.

Esempio

Un semplice esempio di applicazione CGI scritta in Python è il seguente:

```
#!C:\Python26\python.exe

import cgi

print "Content-Type: text/html"
print ""
form = cgi.FieldStorage()
print '<HTML>'
print '  <HEAD>'
print '    <TITLE>Applicazione CGI di prova</TITLE>'
print '  </HEAD>'
print '  <BODY>'
print '    <H1 align = "center">Ecco una semplice applicazione CGI</H1>'
print '    <P ALIGN = "center">'
if form.has_key("a") : print "Adesso il valore del parametro a &egrave;: " +
form["a"].value
else : print "Il parametro a non ha alcun valore."
print '    </P>'
print '    <FORM ACTION = "prova.py" METHOD = "get">'
print '      <P ALIGN = "center">Inserisci il nuovo valore del parametro a:<br>'
print '      <INPUT TYPE = "text" NAME = "a" VALUE="test"/><br>'
print '      <INPUT TYPE = "submit" value = "ok"/></P>'
print '    </FORM>'
print '  </BODY>'
print '</HTML>'
```

6. Le servlet

Che cosa sono le servlet

Una servlet è un componente Java che può essere eseguito da un Web server Java-enabled per l'implementazione di nuovi servizi.

Le servlet costituiscono uno dei meccanismi più usati per la creazione di pagine Web dinamiche e consentono inoltre di realizzare molte altre funzionalità tra cui:

1. Forward delle richieste ad altri server e gestione delle connessioni: le servlet possono implementare i meccanismi di load balancing (a livello applicativo) e partizionamento funzionale di sistemi RAPS. Una servlet può inoltre mantenere più connessioni aperte verso processi client, riducendo il numero di connessioni verso i sistemi di DBMS.
2. Proxy per le applet: una applet eseguita da un browser può, per ragioni di sicurezza, stabilire delle connessioni solo verso il Web server da cui è stata scaricata. Se l'applet richiede l'accesso ad un servizio o DBMS disponibile su una macchina diversa dal Web server, la servlet può realizzare la connessione per l'applet ed assume quindi il ruolo di proxy.
3. Personalizzazione o realizzazione di nuovi protocolli: le servlet consentono di estendere i protocolli Internet esistenti o di realizzare nuovi protocolli. Essenzialmente, ogni protocollo che si basa sul meccanismo richiesta/risposta può essere realizzato attraverso una servlet.

L'esecuzione delle servlet può avvenire all'interno del processo del Web server oppure all'interno di processi specializzati chiamati Servlet Engine o Servlet Container. Se l'esecuzione avviene all'interno del Web server, la comunicazione è più efficiente, mentre l'adozione di Servlet Engine consente di adottare un maggiore numero di tier fisici.

Servlet temporanee e permanenti

In fase di deployment, l'amministratore del sistema può definire un opportuno parametro di configurazione della servlet nel web server; in tal modo la servlet potrà essere definita in 2 modo diversi:

- ♦ **Servlet temporanee**

Le *servlet temporanee* vengono lanciate e fermate per ogni richiesta dei client, come accadeva per gli script CGI. Le servlet temporanee sono un ottimo meccanismo per risparmiare risorse e fornire servizi di uso poco frequente.

- ♦ **Servlet permanenti**

Le *servlet permanenti* vengono mantenute attive finché il Web server che le esegue non viene riavviato e quindi possono, durante il loro ciclo di vita, soddisfare più richieste applicative. Le servlet permanenti vengono utilizzate se il tempo di start-up è elevato, se è necessario realizzare una funzionalità permanente, oppure se è necessario rispondere ai client nel più breve tempo possibile.

L'interfaccia Servlet

Un Web server comunica con le servlet attraverso i metodi definiti nell'interfaccia *javax.servlet.Servlet*, costituita principalmente dai metodi:

1. *init()*: è il metodo che viene invocato quando la servlet viene invocata. Se la servlet è permanente, questo metodo viene invocato solo all'avvio del Web server. Al suo interno, sono definite tutte le operazioni di inizializzazione.
2. *service()*: è il metodo che viene invocato ad ogni richiesta che viene inviata alla servlet. Richiesta e risposta vengono scambiate tramite due oggetti passati come parametro che appartengono alle classi *ServletRequest* e *ServletResponse* rispettivamente. Più invocazioni al metodo *service()* possono essere eseguite contemporaneamente, perciò è necessario garantire che la sua esecuzione sia thread-safe.
3. *destroy()*: è il metodo richiamato per permettere alla servlet di rilasciare le risorse di sistema prima che la sua esecuzione abbia termine. Il Web server tipicamente attende la terminazione di eventuali metodi *service()* prima di invocare *destroy()*; l'esecuzione di un metodo *destroy()* prima della terminazione di un metodo *service()* può generare errore.

La classe `HTTPServlet`

Solitamente però, anziché usare direttamente l'interfaccia server, si preferisce estendere una classe `HTTPServlet`, definita nel package `javax.servlet.http`: tale classe estende implementa l'interfaccia `Servlet` e fornisce un'implementazione del metodo `service()` che inoltra la richiesta HTTP a metodi come `doGet()`, `doPost()` che sono specializzati a elaborare richieste HTTP rispettivamente di tipo GET e POST.

Un semplicissimo esempio di Servlet Java così realizzato è il seguente:

```
index.html
```

```
<html>
  <head><title>Semplice esempio di Servlet Java</title></head>
  <body>
    <h1 align = "center"> Esempio di Servlet Java </h1>
    <form ACTION="ClasseProva" METHOD="POST">
      <p align="center">Scrivi qui il tuo nome:
        <INPUT TYPE="text" NAME="nome"/>
        <br><INPUT TYPE="submit" VALUE="Conferma!"/></p>
    </form>
  </body>
</html>
```

```
ClasseProva.java
```

```
package ste;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class ClasseProva extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {

    private static final long serialVersionUID = 1L;

    public ClasseProva() {
        super();
    }

    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>Errore</title></head>");
        out.println("<body><h1>Errore</h1>");
        out.println("<p>Metodo GET non supportato dalla servlet</p>");
        out.println("</body></html>");
    }

    protected void doPost(HttpServletRequest request,
                         HttpServletResponse response) throws ServletException, IOException {

        String nome = request.getParameter("nome");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Hello!</title></head>");
        out.println("<body>");
        out.println("<h1 align=center>Ciao, " + nome + "</h1>");
        out.println("</body></html>");
    }
}
```

Sandboxing e sicurezza

Il Web server può limitare le operazioni eseguite dalle servlet, per ragioni di sicurezza. Il meccanismo di sicurezza implementato per le servlet è noto come *sandboxing*. Una sandbox è un'area in cui alle servlet viene assegnata un'autorità limitata per accedere alle risorse del server. Per esempio è possibile negare alle servlet l'accesso al file system o alla rete.

Gestione delle sessioni nelle servlet

Le classi di supporto alla gestione delle sessioni che possono essere utilizzate con le servlet sono:

- ♦ *La classe Cookie*

Un cookie deve essere creato ed associato ad una risposta HTTP e verrà fornito in seguito come parte dell'header HTTP di tutte le richieste successive.

Ad esempio, è possibile memorizzare il totale del carrello attraverso il cookie TOTALE CARRELLO. Il metodo service della servlet che realizza la funzionalità di selezione di un item nel carrello inizializzerà i dati della sessione attraverso le seguenti linee di codice:

```
float totaleCarrello = 0;
Cookie c = new Cookie("TOTALE_CARRELLO", Float.toString(totaleCarrello));
response.setContentType("text/html");
response.addCookie(c);
```

Per richiamare il valore di è necessario usare il metodo `getCookies()` della classe `ServletRequest`, che restituisce tutti i cookie del dominio memorizzati sul client, e cercare nell'elenco lo specifico cookie:

```
Cookie c = null;
Cookie cookies[] = request.getCookies();
if(cookies != null) {
    for(int i = 0, n = cookies.length; i < n; i++) {
        c = cookies[i];
        if(c.getName().equals(TOTALE_CARRELLO))sum=Float.parseFloat(c.getAttribute());
    }
}
```

- ♦ *La classe HttpSession*

È più semplice da usare, ma non consente la condivisione delle informazioni attraverso istanze multiple del browser di uno stesso utente. La sessione corrente viene restituita invocando il metodo `getSession()` della classe `HttpServletRequest`, che è una specializzazione di `ServletRequest`. Si possono poi usare i metodi `getAttribute` e `putAttribute()`. Ad esempio:

```
Float totCarrello = (Float)(session.getAttribute("TOTALE_CARRELLO")).floatValue();
totCarrello = totCarrello + newItem.getPrice();
session.setAttribute("TOTALE_CARRELLO", totCarrello);
```

Il servlet container e la fase di deploy

- ♦ *Il servlet container*

L'utilizzo delle servlet presenta un importante vantaggio: il programmatore non deve conoscere il funzionamento di http, perché è il servlet container che lo gestisce. Il servlet container è un middleware che è in grado solamente di fornire dei servizi per aiutare l'esecuzione di una logica applicativa. Il servlet container opera solitamente sulla porta 8080 e viene invocato dall'application server. Di fatto quindi è il servlet container ad invocare le servlet, e non l'application server.

Un esempio di servlet container è TomCat.

- ♦ *L'operazione di deploy*

L'operazione di deploy di una servlet avviene incapsulando in un package tutte le pagine html e le classi compilate che costituiscono la servlet. Tale package viene compresso con l'estensione .war (web archive). Tale package può essere creato automaticamente da tool software come Eclipse (mediante la funzione export). La struttura del war è predefinita:

```
\meta-inf
\web-inf
    web-xml
    classes
    lib
```

La cartella classes contiene tutte le classi compilate che costituiscono la servlet. La cartella lib contiene tutte le librerie che servono per l'esecuzione (es. driver mysql), ma anche eventuali jar usati all'interno delle classi. La cartella lib è quindi una porzione del classpath, le cui classi sono visibili sono nel war.

Nella root sono presenti tutti i file html. Il file web-xml è il descrittore della struttura, scritto in xml: descrive tutte le risorse applicative che ci sono nel war stesso. In tale file, per ogni servlet, si indicano il nome della servlet, il nome della classe che effettivamente implementa la servlet, e infine il contextURL. Una volta creato il file .war, si deve indicare al servlet container il percorso del war: il servlet container si occuperà di copiarlo in una particolare cartella, di decomprimere e di cercare il descrittore identificando così le nuove risorse da mettere a disposizione.

7. JSP

Che cosa sono le JSP

Le JSP (Java Server Pages) sono un'evoluzione della tecnologia servlet, con l'obiettivo di realizzare pagine dinamiche che scorporino la logica di presentazione del quella applicativa. Sostanzialmente, una pagina JSP è una pagina HTML che contiene delle parti aggiuntive per l'esecuzione di codice che genera un contenuto dinamico. Così come per le servlet però, l'esecuzione è server-side, e il browser riceve solo una pagina statica.

Quando l'utente fa richiesta di una pagina JSP, essa viene compilata in una servlet. Le successive richieste invece non richiedono il processo di compilazione (la compilazione cioè avviene solo alla prima richiesta). In caso di modifica della JSP, essa viene automaticamente ricompilato alla prima richiesta.

Elementi di una pagina JSP

Le possibili componenti di una pagina JSP sono:

- ♦ **Direttive**

Le direttive sono istruzioni processate dal JSP Engine al momento della compilazione della pagina JSP in una servlet. Servono per importare opportune librerie di linguaggio oppure per includere file.

Vengono definite all'interno della coppia di tag <%@ e %>.

- ♦ **Espressioni**

Le espressioni sono dei placeholder (segnaposti), che consentono di stampare all'interno della pagina il valore di una variabile o espressione Java, in maniera dinamica durante l'esecuzione. Sono comprese tra la coppia <%= e %>.

- ♦ **Scriptlet**

Sono dei blocchi di codice inseriti all'interno della pagina JSP che consentono di specificare codice piuttosto complesso Vengono definite all'interno dei tag <% e %>.

Sono ammessi i costrutti di base tipici della programmazione strutturata nella sintassi Java ed è possibile accedere agli oggetti predefiniti che descriveremo a breve. È tuttavia consigliato usare poco le scriptlet, per non perdere la separazione tra logica applicativa e di presentazione.

- ♦ **Dichiarazioni**

Sono simili alle dichiarazioni in Java e definiscono variabili che poi vengono utilizzate in espressioni e scriptlet. Vengono definite all'interno di <%! e %>.

- ♦ **Commenti**

Sono stringhe che vengono ignorate dal JSP engine. Possono essere inseriti ovunque nella pagina JSP e vengono racchiusi all'interno di <%-- e --%>. Può essere comunque utilizzata la sintassi dell'HTML racchiudendo il commento nei tag <%!-- e --%>.

- ♦ **Azioni**

Sono un meccanismo che consente di suddividere ulteriormente la logica applicativa dalla presentazione utilizzando componenti, come i JavaBean o gli EJB, e tag library. Una JSP può dichiarare l'uso di un JavaBean attraverso l'action:

```
<jsp:useBean id= nomebean class=package.class />
```

Le espressioni e le scriptlet hanno accesso, oltre che alle variabili, anche ad un insieme di oggetti predefiniti tra cui i principali sono: request, response, session, application e out.

- ♦ **Request**

L'oggetto request viene utilizzato per accedere all'intestazione della richiesta HTTP. I metodi principali dell'oggetto request sono *getParameter()* che consente di accedere al valore di un parametro specificato in una form HTML, ed il metodo *getCookies()* che restituisce l'array dei cookie salvati sul client.

- ♦ **Response**

L'oggetto response viene usato per specificare le proprietà della pagina fornita come risposta, come ad esempio le dimensioni del buffer usato per la trasmissione, oppure per forzare la trasmissione di una pagina creata parzialmente.

- ♦ **Out**

L'oggetto out viene utilizzato per stampare output sulla pagina Web. Ad esempio le due espressioni <%=nomeUtente%> e <% out.print(nomeUtente)%> sono equivalenti.

Sessioni JSP

- ♦ *Uso di session*

Le pagine JSP di default appartengono ad una sessione Web. È possibile avere accesso alle informazioni di stato di una sessione attraverso l'oggetto session. L'oggetto session viene identificato da un ID univoco memorizzato nel client dell'utente attraverso i cookie. Le informazioni di stato vengono identificate attraverso una chiave univoca e memorizzate come oggetti nell'oggetto session, che non può memorizzare tipi primitivi.

Ad esempio, potremo inserire un dato scrivendo:

```
<% Carrello c = new Carrello();
   session.setAttribute("CARRELLO", c); %>
```

oppure, per leggerlo:

```
<% Carrello c = (Carrello) session.getAttribute("CARRELLO"); %>
```

Inoltre, è possibile escludere una pagina da una sessione scrivendo:

```
<%@ page session = "false" %>
```

E si può limitare la durata nel tempo di una sessione mediante l'istruzione:

```
<% session.setMaxInvalidationInterval(int secs) %>
```

- ♦ *Uso di application*

Se si vuole avere una visibilità allargata a tutte le pagine di una stessa applicazione, si può usare l'oggetto application. Questo oggetto mette a disposizione, con ovvi significati, i metodi setAttribute(), getAttribute(), getAttributeNames() e removeAttribute().

Esempio di pagina JSP

Di seguito è riportata una banalissima pagina JSP di esempio:

index.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Semplice esempio di Servlet Java</title>
  </head>
  <body>
    <h1 align = "center"> Esempio di uso di JSP </h1>
    <form ACTION="prova.jsp" METHOD="POST">
      <p align="center">Scrivi qui il tuo nome:
        <INPUT TYPE="text" NAME="nome"/>
      <br><INPUT TYPE="submit" VALUE="Conferma!" /></p>
    </form>
  </body>
</html>
```

prova.jsp

```
<%@ page language="java" info="Prova" %>

<html>
  <head>
    <title>Semplice esempio di Servlet Java</title>
  </head>
  <body>
    <%! String nome; %>
    <% nome = request.getParameter("nome"); %>
    <H1 align="center">Ciao, <%=nome%>!</H1>
  </body>
</html>
```

8. JDBC

Che cosa è JDBC?

Lo standard che viene utilizzato per realizzare la connessione tra programmi Java e sistemi data base è JDBC (Java Database Connectivity). L'API consente l'accesso ai data base attraverso SQL. Concettualmente l'accesso alla base di dati avviene in tre passi:

1. Il programma stabilisce una connessione verso il DBMS;
2. Si esegue l'interrogazione o l'aggiornamento della base di dati tramite esecuzione di statement SQL;
3. Il programma accede ai risultati.

Per instaurare una connessione è necessario caricare innanzitutto il driver del sistema di gestione di base di dati a cui si ha accesso. Ciò è necessario perché JDBC è progettato per essere indipendente dal DBMS usato. Dato che DBMS diversi hanno un comportamento diverso, i dettagli per l'accesso ai sistemi sono implementati nei driver. Ad esempio per accedere ad un DBMS Oracle è necessario eseguire un'istruzione del tipo:

```
Class.forName("sun.jdbc.driver.OracleDriver")
```

La connessione viene poi stabilita creandone un'istanza attraverso l'istruzione:

```
Connection c = DriverManager.getConnection("jdbc:oracle:thin:server:port:Instance",
                                         username, password);
```

L'esecuzione di un comando SQL può avvenire in due modi:

- ♦ **Mediante la classe Statement**

Ad esempio, se il comando è di update (detta c una connessione valida e aperta):

```
Statement stmt = c.createStatement();
stmt.executeUpdate("UPDATE Libro SET prezzo = 25.0
                   WHERE editore = 'Franco Angeli'
                   AND titolo = 'Sistemi Informativi Web'");
};
```

- ♦ **Mediante la classe PreparedStatement**

In questo caso possiamo introdurre degli statement parametrici:

```
PreparedStatement prepareUpdatePrice =
con.prepareStatement("UPDATE Libro
                     SET prezzo = ? WHERE editore = ? AND titolo = ?");
prepareUpdatePrice.setFloat(1, 25.0);
prepareUpdatePrice.setString(2, "Franco Angeli");
prepareUpdatePrice.setString(3, "Sistemi Informativi Web");
prepareUpdatePrice.executeUpdate();
```

In maniera del tutto analoga, l'esecuzione di un'interrogazione può essere eseguita nelle due corrispondenti modalità. Si noti però che occorre poi scorrere l'elenco dei risultati, che vengono restituiti mediante un oggetto della classe ResultSet, la quale mette a disposizione un metodo next: inizialmente, ci si trova alla posizione antecedente rispetto al primo record; dopo la prima invocazione di next, si possono prelevare i dati del primo record, e poi ci si sposta avanti di un record alla volta. Quando si è arrivati alla fine, il metodo next restituisce il valore false.

Ad esempio:

```
ResultSet rs = stmt.executeQuery("SELECT * FROM Sells");
while (rs.next()){
    editor = rs.getString("editore");
    title = rs.getString("titolo");
    price = rs.getFloat("prezzo");
    System.out.println("Editore" + editor + " Titolo " + title
                      + " Prezzo: " + price);
}
```

9. Il pattern MVC

Il pattern MVC

Quando si costruisce un'applicazione interattiva, è sempre opportuno seguire un pattern di progettazione noto come MVC (Model-View-Controller). Tale pattern prevede una scomposizione in 3 parti:

- ◆ **Model**

Rappresenta lo stato e le operazioni disponibili.

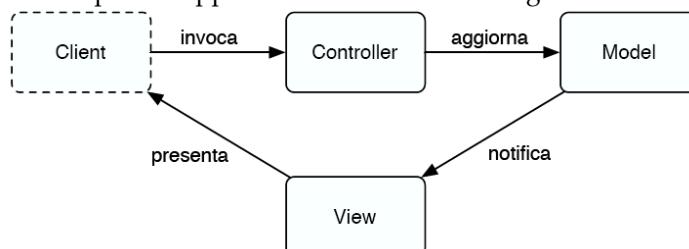
- ◆ **View**

Definisce il modo col quale il modello deve essere presentato all'utente.

- ◆ **Controller**

Interpreta l'azione dell'utente e invoca le operazioni opportune del model.

Lo schema di funzionamento è quindi rappresentabile mediante il seguente schema:



L'utente interagisce con la view. A partire da essa, l'utente può richiedere particolari funzionalità: le richieste vengono quindi invocate dal client (browser) sul controller. Il controller invoca i metodi messi a disposizione del model, il quale modificherà il proprio stato e al termine notificherà al view le modifiche avvenute. A questo punto, la view verrà aggiornata, e quindi le finestre o pagine Web cambieranno il loro aspetto di presentazione.

Implementare il pattern MVC attraverso pagine JSP

In pratica, le architetture che si possono usare con JSP per seguire il modello MVC sono 2:

- ◆ **L'architettura Model 1**

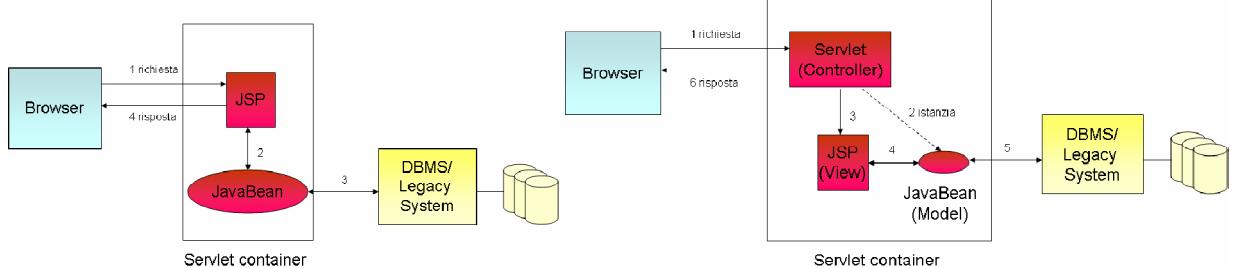
Questa architettura prevede che le pagine JSP siano responsabili della gestione dello stato dell'applicazione. Di conseguenza, non si ha una completa separazione tra logica applicativa e di presentazione, perciò è possibile che le JSP contengano grandi porzioni di codice. Per tale ragione, quest'architettura è consigliata solo se l'applicazione in questione è particolarmente semplice.

- ◆ **L'architettura Model 2**

L'architettura Model 2 prevede che l'utente invii le proprie richieste ad una Servlet, che fa da controller. La Servlet poi, se necessario, instanzia delle nuove classi JavaBean. Inoltre, il controller fa da dispatcher e inoltra (tramite forward) le richieste ricevute ad opportune pagine JSP, fornendo loro i dati che dovranno mostrare in output (ma senza formattarli: di questo si occuperanno le JSP stesse).

In questo modo, si ha una completa suddivisione della logica di presentazione da quella di accesso ai dati: ad esempio, se il web designer dovesse modificare la grafica, dovrebbe solo aggiornare le pagine JSP, e ciò può essere fatto facilmente, perché contengono pochissime porzioni di codice (tipicamente, solo delle espressioni).

Di seguito sono rappresentate, nell'ordine, l'architettura Model 1 e l'architettura Model 2.



Capitolo 10: Progettazione dei WIS

1. La progettazione dei WIS

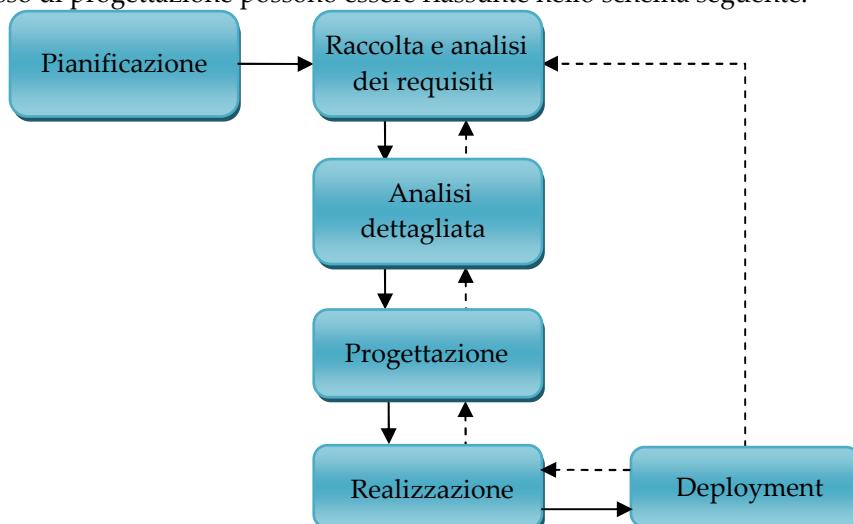
La progettazione come processo

La realizzazione di un sistema informativo richiede un approccio sistematico guidato da un processo di sviluppo ben definito e sistematico, in modo da evitare problemi quali la scarsa scalabilità e manutenibilità. Di conseguenza, la progettazione del WIS viene considerata come un vero e proprio processo, definito in maniera rigorosa, attraverso l'indicazione di:

1. Tutte le fasi che costituiscono la progettazione del WIS e l'ordine in cui si susseguono;
2. Gli elaborati da sviluppare nelle singole fasi di progettazione;
3. Gli incarichi di lavoro dei singoli e del gruppo;
4. I criteri di controllo e valutazione del sistema realizzato.

Le fasi della progettazione

Le fasi del processo di progettazione possono essere riassunte nello schema seguente:



Vediamo adesso di analizzare il significato di tali fasi:

- ♦ **Pianificazione**

Questa fase ha l'obiettivo di produrre uno studio di fattibilità: si individuano i tempi, le risorse, gli obiettivi e i rischi che si hanno nella realizzazione del WIS, e si valuta se è opportuno procedere con le fasi seguenti oppure abbandonare l'intero progetto.

- ♦ **Raccolta e analisi dei requisiti**

In questa fase si definiscono gli elementi principali che costituiranno il sistema finale e le funzionalità che dovranno essere messe a disposizione dell'utente.

- ♦ **Analisi dettagliata**

Questa fase serve a definire le funzionalità e i limiti del sistema, ma anche (a grandi linee) i moduli software che comporranno il WIS, ponendo l'accento soprattutto sulla divisione tra accesso ai dati, logica applicativa e presentazione.

- ♦ **Progettazione**

La progettazione è la fase in cui si arriva a ottenere il dettaglio del sistema pronto per essere realizzato.

- ♦ **Realizzazione**

La realizzazione è la fase in cui il WIS viene realmente implementato, utilizzando uno specifico linguaggio di programmazione e una particolare piattaforma.

- ♦ **Deployment**

In questa fase, tutti i componenti vengono installati e il WIS viene messo in esercizio.

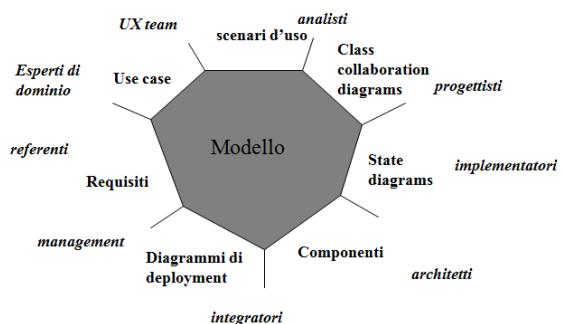
Come si nota, si ha un ciclo che coinvolge alcune fasi del processo: nel caso in cui sorgano incongruenze oppure emergano nuove esigenze, il ciclo viene ripetuto. Si nota inoltre che non è specificata una fase di test, perché si suppone che in ogni fase venga effettuata un'attività di verifica.

Professionalità coinvolte nella progettazione del WIS

Le professionalità coinvolte nella progettazione di un WIS sono molteplici e non solo informatiche: si ha infatti particolare attenzione anche alla progettazione grafica e alla redazione dei contenuti.

Inoltre, gli attori coinvolti sono:

1. Il management dell'organizzazione che ha commissionato il WIS;
2. I referenti o stakeholder, interni o esterni all'organizzazione.
3. Esperti di dominio, che collaborano alla definizione dei requisiti del WIS.
4. Analisti, progettisti, implementatori, architetti del software, integratori.



Lo schema riportato riassume le professionalità coinvolte nella progettazione del WIS, insieme ai prodotti che realizzano.

Qualità dei WIS

I principali fattori che caratterizzano la qualità dei WIS sono:

- ♦ **Navigabilità**
I collegamenti tra le varie pagine di un sito Web devono essere progettati per rendere facile il passaggio da una pagina a quelle funzionalmente collegate ad essa.
- ♦ **Accessibilità**
Le informazioni devono essere accessibili a tutti i tipi di browser.
- ♦ **Usabilità**
È la facilità per gli utenti nell'utilizzare il sito (ad esempio, nell'interpretare i pulsanti).
- ♦ **Leggibilità**
Le informazioni devono essere mostrate in modo adeguato (per presentazione e contenuti).
- ♦ **Affidabilità**
Deve essere garantito il funzionamento del sistema.
- ♦ **Manutenibilità**
Il sito deve essere facilmente mantenibile e deve essere possibile aggiornarlo senza difficoltà.
- ♦ **Compatibilità e interoperabilità**
Deve essere possibile integrare e far interoperare tra loro diversi sistemi per fornire i servizi Web.
- ♦ **Sicurezza**
Le informazioni riservate devono essere protette e l'accesso ai servizi deve essere consentito solo agli utenti autorizzati.
- ♦ **Efficienza**
Le risorse usate dal sistema devono essere contenute e i tempi di risposta non eccessivi.

Per ogni singolo fattore di qualità, si individuano poi delle variabili significative per la loro valutazione. Nel caso in cui tali qualità siano difficilmente misurabili, si procede con indagini o questionari. Si deve inoltre tener conto che diverse qualità hanno livelli di importanza diversa a seconda del punto di vista che si assume (ad esempio, dell'utente, del gestore, dello sviluppatore o del committente).

Il metodo di Conallen

Il processo di sviluppo del sistema informativo viene eseguito adottando una metodologia di progettazione particolare, che è stata definita da Conallen per la progettazione di applicazioni basate sul Web, e che viene qui ripresa con alcune modifiche. Questo modello di progettazione, detto WAE (Web Application Extension), è basato sull'uso di UML ed è caratterizzato dal fatto di essere guidato dai casi d'uso, incentrato sull'architettura, e di tipo iterativo ed incrementale (diversamente dal classico progetto del software a cascata).

Andiamo quindi, a partire dal prossimo paragrafo, a definire nel dettaglio tutti i passi che si devono seguire per l'utilizzo di questa metodologia.

2. La fase di pianificazione

La fase di pianificazione

La fase di pianificazione può a sua volta essere scomposta in fasi distinte:

1. Analisi dell'azienda e del problema;
2. Analisi del dominio;
3. Vision;
4. Realizzazione del piano di progetto.

Di queste attività, le prime 2 possono essere eseguite in parallelo.

Analisi dell'azienda e del problema

In questa fase, un gruppo di analisi cerca di comprendere le attività dell'azienda legate al WIS che si intende realizzare, mediante l'interazione con gli stakeholder e l'analisi della documentazione dell'azienda stessa. Questa fase è essenziale per poter definire con precisione gli obiettivi dell'azienda. Durante l'analisi dell'azienda, si deve cercare di non essere influenzati dalla percezione degli stakeholder.

Analisi del dominio

Nell'analisi del dominio ci si sofferma su:

1. I concetti rilevanti del dominio applicativo, rappresentati mediante un diagramma ER oppure mediante un diagramma delle classi;
2. I processi aziendali allo stato attuale, rappresentati con diagrammi di attività.

Si realizza inoltre un glossario dei termini contenuti nei diagrammi, con le loro definizioni. In sostanza quindi in questa fase si realizza un modello dell'azienda e del dominio.

Vision o sviluppo degli obiettivi

♦ *Che cos'è la fase di vision*

In questa fase si utilizzano come punto di partenza i risultati delle due fasi precedenti. Il risultato di questa fase è un documento, detto appunto documento di vision, che esprime il contesto e gli obiettivi dell'intero progetto e che è solitamente usato come base per decidere se finanziare oppure no il progetto.

Il documento di vision delinea in maniera molto approssimativa quale sarà la soluzione del problema, mostrando gli elementi di innovazione, le architetture che si intendono usare e, in generale, la direzione che si intende seguire, senza però entrare nel dettaglio tecnologico.

♦ *Struttura del documento di vision*

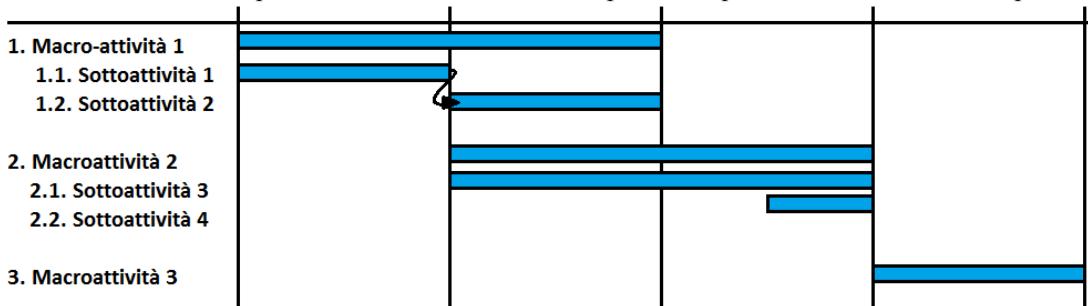
In particolare, il documento di vision si articola nei seguenti punti:

1. Introduzione: si identificano i referenti e le ragioni per le quali realizzare il sistema informativo.
2. Background: si individuano i sistemi preesistenti e il contesto aziendale.
3. Requisiti e funzionalità generali: si descrive brevemente cosa dovrà fare il sistema informativo.
4. Requisiti architetturali: si definisce a grandi linee l'architettura del sistema (ad esempio, si dice che si tratterà di un WIS, si stabilisce da quali browser dovrà essere accessibile il sistema, ma anche quali sono i sottosistemi che lo costituiscono, o le basi di dati da usare, ...).

Il piano di progetto

Questa fase è quella di pianificazione vera e propria, nella quale si stabiliscono quali sono le risorse da mettere a disposizione (in termini economici e di tempo) e quale sarà il prezzo finale del sistema informativo. A tale scopo, viene sfruttata la metodologia WBS, che consiglia di suddividere il problema in sottoprodotto da realizzare, e di individuare le risorse da utilizzare per ogni sottoproblema.

In particolare, si realizzano dei diagrammi, detti *diagrammi di Gantt* (o *cronoprogrammi*), che identificano delle macro-attività, ciascuna scomposta in sottoattività. Per ogni attività o sottoattività vengono definiti l'istante di inizio e l'istante di fine, con l'aggiunta di eventuali dipendenze esplicite (evidenziate mediante frecce): alcune attività infatti possono essere iniziate solo a patto che prima ne siano state completate altre.



Si noti che è sempre importante prevedere come macro-attività finale quella di formazione di coloro che saranno gli utilizzatori del sistema informativo prodotto.

Oltre ai tempi, bisogna individuare le scadenze (dette anche *milestone*) e i costi, che possono essere:

- ♦ *Costi di progetto*
Sono i costi che devono essere affrontati per poter portare a termine il lavoro.
- ♦ *Costi di esercizio*
Sono i costi necessari al mantenimento del sistema informativo realizzato.

3. La fase di raccolta e analisi dei requisiti

La fase di raccolta e analisi dei requisiti

La fase di raccolta e analisi dei requisiti può essere suddivisa in due "sotofasi":

- ♦ *Raccolta dei requisiti*
Si parte dai documenti prodotti durante la fase di pianificazione, e si cerca di approfondire i requisiti del sistema informativo.
- ♦ *Specificazione dei requisiti*
A questo punto, i requisiti, fino ad ora espressi solamente in linguaggio naturale, devono essere espressi mediante notazioni formali o semiformali, come diagrammi UML o schemi ER.

L'obiettivo complessivo della fase è quello di dare una descrizione non ambigua del sistema, che risulti abbastanza completa ma al contempo che non entri troppo nel dettaglio, perché altrimenti si rischia di prolungare troppo nel tempo la durata di questa fase e di produrre una documentazione troppo pesante e voluminosa.

La raccolta dei requisiti

- ♦ ***Da dove provengono i requisiti?***

Questa fase viene svolta a partire da:

1. La documentazione preesistente, soprattutto il documento di visione e i modelli di dominio;
2. Interviste agli stakeholder e a gruppi di utenti selezionati, che indicano quali devono essere le funzionalità desiderate e come si desidera che tali funzionalità si presentino dal punto di vista dell'utente.

- ♦ ***Come vengono documentati i requisiti iniziali?***

I requisiti che vengono espressi al termine di questa fase vengono espressi mediante un insieme di punti numerati, ciascuno con eventuali sotto-punti (numerazione gerarchica). Ogni requisito deve esprimere, in linguaggio naturale, una proprietà verificabile del sistema (ad esempio, non "il sistema deve avere tempi di risposta accettabili", ma "il sistema, con 10 utenti collegati, deve avere tempi di risposta inferiori a 10 s"). Inoltre, risulta utile associare dei livelli di priorità ad ogni requisito.

La numerazione dei requisiti consente di garantire la *tracciabilità* dei requisiti: nelle fasi successive infatti viene associato ai vari elementi dei modelli formali almeno un requisito, in modo tale che a fronte della modifica dei requisiti (che nei WIS sono particolarmente instabili) si possano individuare subite le conseguenze che ne derivano.

- ♦ ***Classificazione dei requisiti***

I requisiti possono essere di vario tipo:

1. **Requisiti funzionali**

I requisiti funzionali descrivono le funzionalità che dovranno essere inserite nel sistema. Tali requisiti descrivono quindi le operazioni che gli utenti potranno eseguire con il sistema (specificando quali utenti possono eseguirli), ma anche alcune funzionalità interne e le interazioni del sistema informativo con altri sistemi ad esso esterni.

2. **Requisiti non funzionali**

I requisiti non funzionali possono essere ulteriormente classificati in:

- 2.1. ***Requisiti di business (business rules)***

Sono requisiti derivati dai processi aziendali, che comprendono regole di business o riferimenti a documenti che descrivono gli standard aziendali. Ne sono ad esempio, i requisiti che riguardano le politiche di sicurezza.

- 2.2. ***Requisiti architetturali***

Questi requisiti specificano dei limiti tecnologici e riprendono alcune importanti qualità dei WIS che abbiamo precedentemente descritto, come:

1. L'usabilità (es.: massimo numero di click per raggiungere una funzionalità);
2. La robustezza e l'affidabilità;
3. La sicurezza (chi ha accesso al sistema, l'autentizzazione, la crittografia, ...);
4. Le performance.

Esistono inoltre altri importanti vincoli riguardanti la piattaforma da utilizzare, i browser da cui poter accedere al WIS, i set di caratteri ammissibili, i requisiti hardware minimi, ...

I requisiti architetturali vengono espressi in un documento, detto SAR (System Architectural Requirements), che considera:

- a) Il punto di vista dei requisiti generali (funzionalità osservabili, browser, ...).
- b) Il punto di vista del design (piattaforme, middleware, sistemi software da usare, ...).
- c) Il punto di vista della realizzazione (requisiti hardware, librerie, namespaces, ...).
- d) Il punto di vista del testing (misure di qualità e prestazioni, piano di testing, ...).

La specifica dei requisiti

- ♦ **In cosa consiste questa fase?**

La fase di specifica dei requisiti prevede che si riprendano tutti i requisiti appena raccolti, che potranno anche presentare delle incongruenze ed essere a livelli di dettaglio molto diversi tra loro, integrandoli poi in un insieme di requisiti coerenti che costituiranno la base per le fasi successive.

- ♦ **Quali modelli usare?**

In particolare, tutti i requisiti vengono in questa fase espressi mediante notazioni formali o semiformali:

- a) Per quanto riguarda i requisiti funzionali, si utilizza UML, mediante:

1. Use case diagram;
2. Activity diagram o sequence diagram per la rappresentazione di scenari;
3. Class diagram (eventualmente sostituiti da schemi ER) per la rappresentazione dei dati;
- b) Per quanto riguarda invece i requisiti non funzionali, si usa la notazione UX (User eXperience), che consente di modellare l'esperienza dell'utente nell'uso del sistema informativo.

- ♦ **Come procedere?**

Di conseguenza, è opportuno seguire, in generale, i seguenti passi per il completamento della fase di specifica dei requisiti (passi che verranno descritti dettagliatamente nei prossimi paragrafi):

1. Individuazione di tutti gli attori coinvolti;
2. Identificazione dei casi d'uso;
3. Realizzazione e documentare lo use case diagram (con gli attori e gli use case già individuati);
4. Rappresentazione degli scenari (mediante activity diagram e sequence diagram);
5. Progettazione di dati (mediante un class diagram o, più spesso, uno schema ER).
6. Realizzare lo schema UX.
7. Rappresentazione della dinamica dell'interazione col sistema.

La specifica dei requisiti: le sotto-fasi fino alla realizzazione degli use-case diagram

Soffermiamoci adesso sui primi passaggi che costituiscono la specifica dei requisiti, che ci portano ad ottenere gli use case diagram:

- ♦ **Individuazione degli attori**

Per prima cosa, è opportuno stilare un elenco di tutti gli attori coinvolti, ovvero di tutti:

1. Gli utenti del sistema, ovvero le persone che useranno il sistema o che dovranno in qualche modo interagire con esso per fornire dei servizi (*utenti che svolgono attività di back-end*);
2. Gli altri sistemi informativi con i quali il sistema stesso deve interagire.

Le risorse utilizzate per la realizzazione del sistema (ad esempio, la base di dati) non sono considerate attori del sistema, perché sono parte del sistema stesso).

- ♦ **Identificazione dei casi d'uso**

Ogni caso d'uso è un servizio fornito dal sistema informativo agli attori che interagiscono con esso. I casi d'uso vengono facilmente individuati a partire dai requisiti funzionali del sistema, perché un caso d'uso corrisponde ad un verbo che indica un'azione (es.: ordinare, spedire, ...).

- ♦ **Disegnare il diagramma dei casi d'uso**

A questo punto, possiamo procedere con la realizzazione dei diagrammi dei casi d'uso, mediante l'applicazione delle regole che abbiamo già descritto nel capitolo 6.

- ♦ **Documentare i casi d'uso**

Oltre a disegnare il diagramma dei casi d'uso, è sempre opportuno realizzare una documentazione da associare ad esso. Tale documentazione può essere realizzata mediante altri diagrammi UML (es.: sequence diagram o activity diagram) oppure come semplice testo libero. In ogni caso, lo scopo è quello di specificare per ogni servizio quali funzionalità svolge e quali sono le modalità mediante le quali interagisce con il sistema. Di solito, per ogni use case si indicano:

1. Il nome che identifica lo use case;
2. La descrizione dello use case;
3. Gli attori che usano le funzionalità dello use case o sui quali esso si poggia per fornirle;
4. I casi d'uso con i quali lo use case è correlato;
5. Le precondizioni (che devono essere verificate nel momento in cui si interagisce con lo use case);
6. Le postcondizioni (che devono essere verificate quando termina l'esecuzione dello use case);
7. Altri diagrammi per meglio definire lo use case.

La specifica dei requisiti: la fase di rappresentazione degli scenari

A questo punto, la metodologia prevede che si “esploda” ogni caso d’uso in un diverso scenario, rappresentato mediante un diagramma delle attività oppure mediante un sequence diagram.

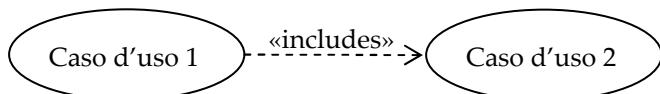
- ♦ *Realizzazione dell’activity diagram*

Anche per quanto riguarda gli activity diagram, sono già state specificate nel dettaglio tutte le regole di stesura, nel capitolo 6. Come accennato, si deve realizzare un activity diagram per ogni scenario.

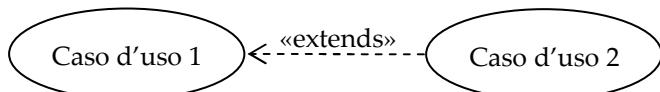
In particolare, è importante rappresentare anche nell’activity diagram la suddivisione tra le attività svolte dall’attore (o dagli attori) e quelle svolte invece dal sistema, perciò si suddivide il diagramma in due diverse lane. L’inizio dell’attività è posto nel lane dell’attore se è quest’ultimo ad avviare l’esecuzione dell’attività stessa; altrimenti, è posto nel lane del sistema.

Nell’activity diagram è possibile che vengano richiamate attività non elementari, ma corrispondenti ad altri use case diagram. In tal caso, si associa a questa attività un commento, contenente il nome dello use case corrispondente. Si noti in particolare che:

1. Se un caso d’uso è legato ad un altro mediante lo stereotipo «includes» (il caso d’uso 1 richiede necessariamente l’esecuzione del caso d’uso 2), allora nell’activity diagram del caso d’uso 1 si dovrà necessariamente eseguire l’attività associata al caso d’uso 2.



2. Se un caso d’uso è legato ad un altro mediante lo stereotipo «extends» (il caso d’uso 1 richiede a volte l’esecuzione del caso d’uso 2), allora nell’activity diagram del caso d’uso 1 si dovrà avere la possibilità di eseguire l’attività associata al caso d’uso 2, ma al contempo dovrà essere previsto un cammino alternativo in cui tale attività non viene eseguita.



- ♦ *Realizzazione dei sequence diagram*

Mentre un activity diagram individua contemporaneamente tutti i possibili scenari che si possono verificare, un sequence diagram rappresenta un solo scenario (cioè uno solo dei possibili percorsi individuabili all’interno dell’activity diagram). Di conseguenza, una volta realizzato l’activity diagram associato ad uno use case, si procede associando ad esso più sequence diagram (lo scenario principale, più eventuali scenari alternativi), secondo le regole che abbiamo già visto nel capitolo 6.

A questo livello, il sequence diagram presenterà sempre 2 attori: il sistema e l’attore vero e proprio. Nelle fasi successive, il sequence diagram viene poi raffinato ulteriormente, scomponendo il sistema nelle sue varie componenti. Per ora però è interessante solamente sapere, in maniera intuitiva, il modo in cui il sistema si comporta nei vari scenari.

La specifica dei requisiti: la fase di definizione dei dati

A questo punto, si può realizzare uno schema (solitamente di tipo ER, oppure un diagramma delle classi) che consenta di definire i dati del sistema informativo. Questo schema è ottenuto a partire dai requisiti del sistema, facendo attenzione soprattutto ai sostantivi che vengono usati. Noi non ci soffermiamo sulle modalità mediante le quali tale schema viene prodotto, perché sono già state abbondantemente discusse durante il corso di Basi di Dati.

La specifica dei requisiti: la fase di definizione dell'interazione con l'utente

Questa fase ha l'obiettivo di rappresentare la navigazione del sito e le pagine principali, con l'indicazione della sequenza delle interazioni previste. Si realizza perciò un diagramma che mostra tutte le schermate del sistema informativo, e i possibili passaggi da una schermata ad un'altra. Si dice anche che tali diagrammi mettono in evidenza l'*esperienza dell'utente*.

A tale scopo, si adotta il modello UX (*User eXperience model*), proposto da Conallen: si tratta di un diagramma delle classi, arricchito con opportuni stereotipi:

- ◆ ***Lo stereotipo «screen»***

Rappresenta una schermata presentata all'utente nella navigazione del sito. Il nome della classe è il titolo della schermata. Una classe con lo stereotipo screen presenta:

1. ***Attributi***

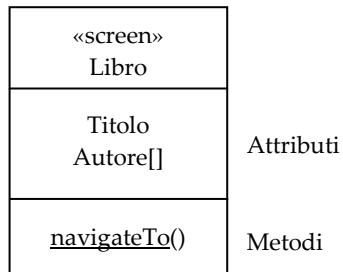
Si ha un insieme di attributi, che rappresentano le informazioni dinamiche (e non quelle statiche!) mostrate all'interno della pagina.

2. ***Metodi***

Si ha poi un insieme di metodi, che rappresentano le azioni che l'utente può compiere a partire da quella schermata, e che si traducono necessariamente in una modifica dello stato dell'applicazione (di solito mediante l'attivazione di un percorso di navigazione verso un'altra schermata). Va messo in evidenza che i metodi qui indicati seguono l'ottica dell'utente, ma potrebbero poi non essere coincidenti con i metodi che effettivamente sarà necessario implementare.

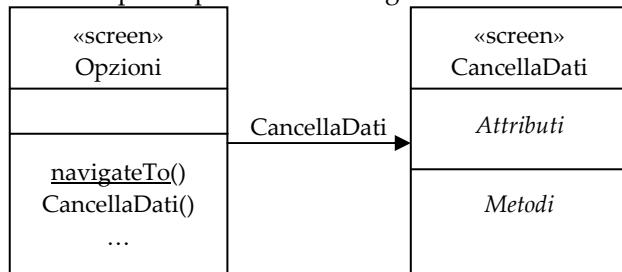
Tra questi metodi, compare sempre il metodo `navigateTo()`, che è il metodo usato per instanziare la schermata stessa. Siccome è statico, il suo nome è sottolineato.

Ad esempio, potremo così rappresentare una schermata che mostra le informazioni di un libro:



Si noti che le parentesi quadre sono dovute al fatto che si suppone che un libro abbia uno o più autori. Per indicare che esiste un link che porta da una schermata 1 ad una schermata 2, si disegna una freccia che parte dalla schermata 1 e arriva alla schermata 2, accompagnata da un'etichetta, che corrisponde solitamente al nome del metodo della prima screen che consente l'attivazione di quel percorso di navigazione. In ogni caso, sono anche ammesse associazioni non orientate.

La figura seguente mostra un esempio di percorso di navigazione:



L'etichetta può anche essere del tipo "Ok" oppure "Errore", per indicare che un certo percorso di navigazione viene attivato quando un'operazione ha avuto successo o quando invece è terminata in modo fallimentare.

Possiamo inoltre specificare dei *landmark*, indicati accanto al nome della classe, che consentono di caratterizzare meglio la schermata. I landmark che useremo sono 2:

1. Landmark \$

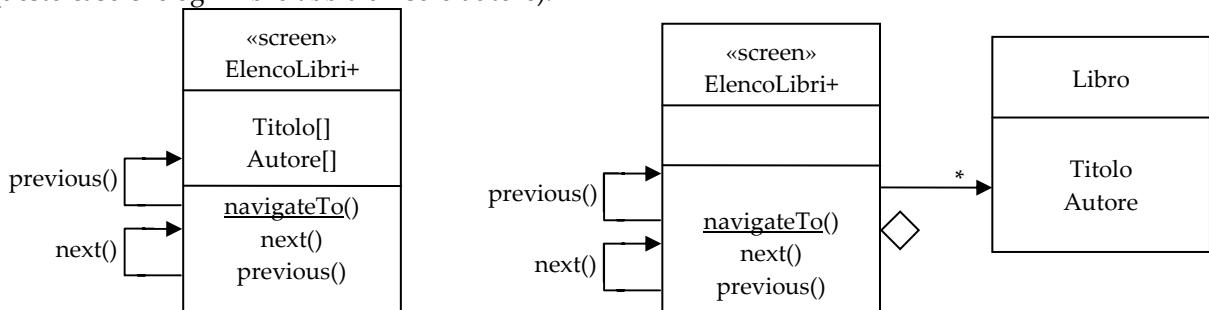
Questo landmark indica che la pagina può essere raggiunta tramite una barra di navigazione generale, a partire da qualsiasi altra schermata del sistema informativo (solitamente, appartiene a questa categoria l'home page del sito). Per questa pagina si può perciò omettere di rappresentare il percorso di navigazione che ha origine da tutte le altre pagine del sistema informativo.

Inoltre, una pagina con \$ può essere instanziata dall'utente mediante la digitazione di un comando (ad es, l'inserimento dell'indirizzo nel browser), mentre le altre schermate sono raggiungibili solo mediante i percorsi di navigazione indicati nello UX.

2. Landmark + (scrollable)

Questo simbolo indica che si tratta di una pagina *multi-istanza*: si ha una sequenza di schermate, tutte con la stessa struttura. Di solito, lo si usa quando si devono mostrare dati che sarebbero troppo lunghi per essere mostrati su un'unica pagina, e perciò vengono spezzati su più pagine contenenti ciascuna un certo numero di dati (come fa Google con la separazione in pagine dei risultati di una ricerca). In questo caso, devono essere presenti i metodi next() e previous() e i corrispondenti percorsi di navigazione, che in realtà non sono altro che semplici autoanelli.

Per rappresentare i dati dinamici è possibile far ricorso, oltre che agli attributi, anche all'associazione (mediante aggregazione: l'oggetto che fa parte della schermata esiste indipendentemente dalla schermata stessa) con delle classi non stereotipate. Questo si usa soprattutto quando la stessa pagina rappresenta un insieme di elementi dello stesso tipo, perché tale associazione consente di indicare la cardinalità multipla. Ad esempio, le due rappresentazioni seguenti sono equivalenti (si suppone in questo caso che ogni libro abbia un solo autore):



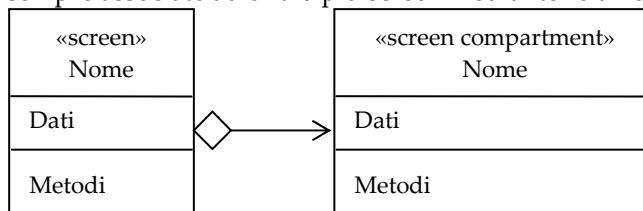
Tuttavia, la seconda rappresentazione è da preferirsi, perché:

1. Mette in evidenza che nella schermata non sono mostrate semplicemente due sequenze, una di titoli e una di autori, ma consente di evidenziare che la screen contiene un elenco di libri, ciascuno dei quali ha un titolo e una autore.
2. Nel caso in cui la schermata mostri l'elenco dei libri con un massimo di 10 libri per pagina, è possibile indicare il valore 10 al posto dell'asterisco, fornendo così un ulteriore dettaglio.
3. Se altre schermate mostrano i dati di tipo Libro, è possibile riusare la classe Libro, legandola anche con altre classi con lo stereotipo screen.

Si noti che le classi non stereotipate non corrispondono a tabelle del database e che nello UX model non è possibile rappresentare legami tra classi di questo tipo.

♦ Lo stereotipo «screen compartment»

Si tratta di una parte di schermata (ad esempio, la schermata può essere composta da banner oppure parti comuni a più pagine che si scorporano in classi a parte). Per definizione, queste classi non hanno vita propria, ma sono sempre associate ad una o più screen mediante relazione di composizione:



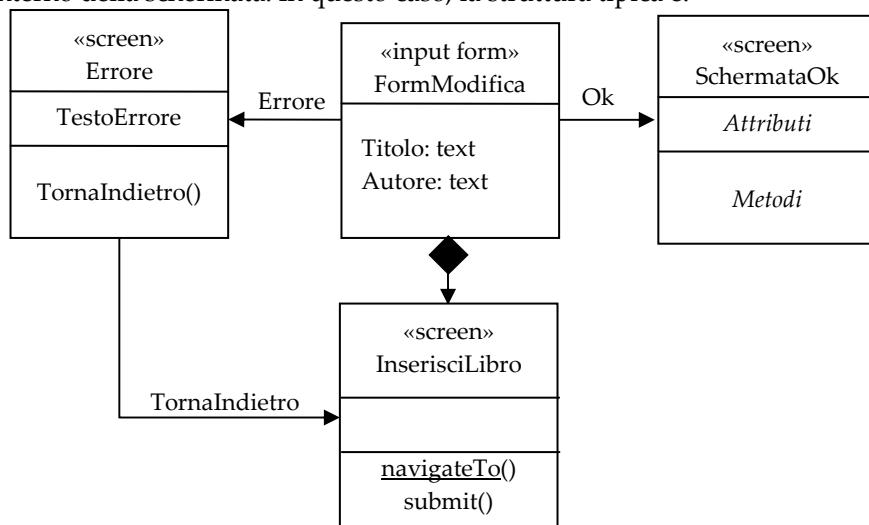
Per il resto, valgono per le screen compartment tutte le considerazioni che abbiamo fatto per le screen. In ogni caso, useremo molto raramente questo stereotipo.

- ♦ **Lo stereotipo «input form»**

Una input form (o maschera) è un modulo di inserimento dati, che consente ad un utente di fornire informazioni al sistema informativo. Le input form non hanno metodi, ma solo un elenco di attributi che corrispondono ai campi che l'utente può riempire mediante l'input form stessa. Per ogni attributo si indica anche il tipo (ad es.: text, radio, ...). Una input form non può essere legata a entità

Una classe con lo stereotipo input form deve sempre essere collegata ad una screen, mediante:

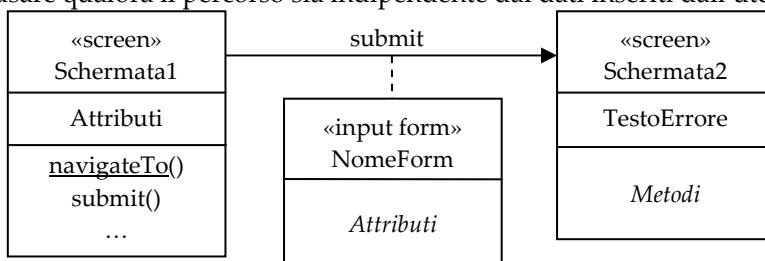
- a) Un'associazione di composizione: questa rappresentazione è opportuna se il percorso di navigazione cambia a seconda dell'esito dell'elaborazione delle informazioni fornite dall'utente. Si noti che si tratta di un'associazione di composizione e non di aggregazione: la form non può esistere se non all'interno della schermata. In questo caso, la struttura tipica è:



La pagina di errore può in alcuni casi contenere un testo dinamico (come nel caso mostrato in figura), oppure può contenere un testo statico (in tal caso, la screen non avrà attributi). Tipicamente poi la screen avrà un metodo per ritornare alla schermata InserisciLibro. La SchermataOk potrebbe poi essere, ad esempio, la pagina che mostra l'elenco aggiornato dei libri (pagina già riportata in un precedente esempio).

In fase di progettazione, si definisce spesso per prima cosa il percorso corretto, e poi si passa a definire il percorso che si ha in caso di errore.

- b) Un legame (individuato da un segmento tratteggiato) con un percorso di navigazione. Questa modalità si può usare qualora il percorso sia indipendente dai dati inseriti dall'utente.



In questo tipo di rappresentazione, la form appartiene sempre alla schermata di origine dell'associazione.

In ogni caso, la screen alla quale la form è associata deve avere, tra i propri metodi, anche un metodo submit(): tale metodo, che può eventualmente avere un altro nome (anche perché altrimenti sorgerebbero problemi di disambiguazione quando una screen contiene più form), è il metodo che viene attivato quando l'utente preme il pulsante di invio dei dati inseriti.

Se la screen a cui è collegata l'input form ha degli attributi con gli stessi nomi della form (oppure è legata ad un'entità con attributi aventi gli stessi nomi della form), allora si indica che i campi della form vengono automaticamente compilati. Una input form invece non può essere direttamente associata ad un'entità.

La specifica dei requisiti: rappresentazione della dinamica dell'interazione

La rappresentazione mediante uno schema che rispetti il modello UX fornisce una visione statica del sistema: se si vuole comprendere la dinamica delle interazioni con il sistema in analisi, si possono scegliere due strade alternative:

- ◆ **Sequenze di schermate**

In questo caso, si presenta una semplice successione di immagini, che mostrino la sequenza di interazione basata su dati di esempio (si mostra così un mock up del sistema informativo).

- ◆ **Diagrammi di interazione**

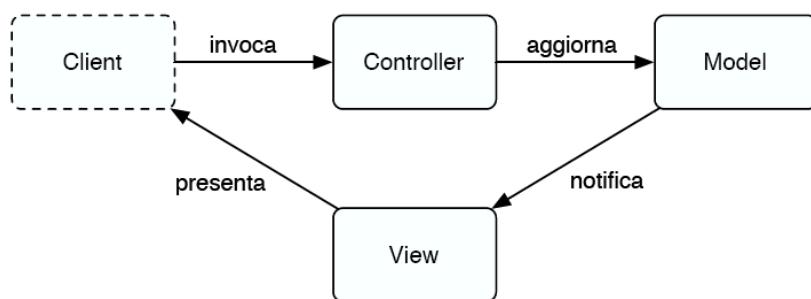
In alternativa, è possibile riprendere i diagrammi di sequenza descritti in precedenza, e raffinarli ulteriormente. In questo caso, il sistema viene esploso nelle singole schermate che lo costituiscono, e si mostra la sequenza di invocazione dei metodi appena definiti all'interno dello UX.

Si noti che in ogni pagina, il primo metodo ad essere invocato deve essere il metodo `navigateTo`. Inoltre, l'utente può invocare direttamente solo le pagine identificate dal landmark `$`: per poter inviare dei messaggi alle altre pagine, dovrà prima essere invocato (da parte di un'altra pagina, e mai direttamente dall'utente, se non si ha il `$`) il metodo `navigateTo` su tale pagina. Naturalmente, i messaggi corrispondono sempre a metodi che sono stati indicati all'interno del precedente modello UX.

4. La fase di analisi dettagliata

In che cosa consiste la fase di analisi dettagliata?

La fase di analisi dettagliata ha l'obiettivo principale di separare gli elementi relativi alla presentazione da quelli che costituiscono la logica applicativa e i dati necessari al WIS. Ci si avvicina quindi al pattern MVC, che i cui principi sono riassunti nello schema seguente:



La parte di presentazione qui però accantonata, e verrà ripresa solamente nella fase di design: ci si concentra perciò sugli aspetti riguardanti la business logic e i dati, adottando non più il punto di vista dell'utente, ma quello del sistema informativo.

Questa fase poi, insieme a quella successiva (di presentazione), consente di ottenere il modello completo del sistema informativo che dovrà poi essere realizzato, e che implementa tutte le funzionalità richieste. Si noti che la fase di analisi dettagliata potrebbe anche iniziare prima del completamento della raccolta dei requisiti, e può talvolta avvenire in parallelo rispetto alla progettazione.

I package

Per facilitare la progettazione da parte di più gruppi di progettisti e mantenere la consistenza tra le diverse parti del progetto, il sistema informativo viene suddiviso in packages, secondo le regole di UML. Ogni singolo package (insieme a tutti gli oggetti ad esso appartenenti) è sotto la responsabilità del responsabilità di una persona, che viene detta appunto *responsabile del package*.

È possibile anche evidenziare delle relazioni di dipendenza tra i package, mediante dei segmenti orientati rappresentati con un tratteggio. Tali segmenti indicano che un package (quello sorgente) usa classi definite nell'altro (il package di destinazione). In ogni caso, è opportuno che le gerarchie che vengono così a crearsi siano poco profonde.

Gli obiettivi che si devono perseguire nella suddivisione in package sono:

- ◆ **Comprensibilità**

Il package deve contenere classi riguardanti la stessa area di interesse.

- ◆ **Coesione**

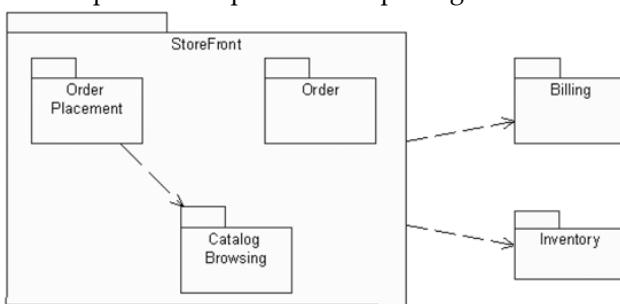
Gli elementi nel package devono essere fortemente collegati tra loro.

- ◆ **Accoppiamento debole**

I collegamenti tra elementi di package diversi devono essere limitati.

Si noti che questa scomposizione verrà mantenuta in tutte le fasi successive (compresa la realizzazione).

La figura seguente mostra un esempio di scomposizione in package.



La progettazione dei dati

Si passa poi a raffinare il modello concettuale dei dati che era stato tracciato nelle fasi precedenti, adottando le tradizionali tecniche della progettazione delle basi di dati. A questo punto, la struttura dei dati risulterà consolidata, e perciò alla progettazione concettuale seguiranno anche quella logica e fisica.

Il diagramma di analisi: schema BCE

Allo scopo di dividere tra loro i componenti appartenenti ai 3 diversi livelli logici dell'applicazione, si realizza uno schema che segue il modello detto BCE (*Boundary-Control-Entity*). Tale modello è un particolare modello delle classi UML, che presenta tre stereotipi fondamentali (quelli che ritroviamo nell'acronimo con cui viene indicato il modello BCE stesso); in ogni BCE ci deve essere almeno una classe che seguia ciascuno dei tre stereotipi che andiamo ora a presentare. Gli stereotipi usati dal BCE sono:

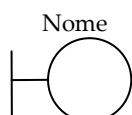
- ♦ **Lo stereotipo «boundary»**

Questo stereotipo serve per rappresentare in maniera molto sintetica l'interazione con l'utente (boundary significa letteralmente confine). Tipicamente, una classe boundary corrisponde ad un certo insieme di screen presenti nel modello UX. In particolare, una regola pratica spesso valida (ma non sempre) prevede che si inserisca nel BCE un boundary per ogni use case.

Le classi boundary non possono avere attributi. I loro metodi invece sono principalmente di due tipi:

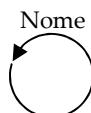
1. I metodi che erano presenti nelle schermate alle quali il boundary corrisponde, e che consentono la navigazione tra le varie schermate. Solitamente si omettono però i metodi next() e previous(). Inoltre, se le screen hanno metodi come submit(), è opportuno che essi vengano rinominati. Tali metodi vengono poi invocati dall'utente.
2. Una serie di metodi che consentono la visualizzazione dell'output all'utente, il cui nome è solitamente del tipo showNomeSchermata(). Questi metodi verranno invocati dal sistema.

Una classe boundary può essere rappresentata, oltre che come normale classe con l'indicazione dello stereotipo usato, anche mediante il simbolo seguente:



- ♦ **Lo stereotipo «control»**

Lo stereotipo control rappresenta un elemento della logica applicativa. Anche in questo caso, si può adottare un simbolo sintetico per rappresentare un oggetto di tale classe:



Gli oggetti di tipo control gestiscono lo stato degli oggetti applicativi utilizzati per fornire le funzionalità descritte attraverso i boundary. Hanno inoltre il compito di rappresentare eventuali comportamenti dipendenti dallo stato e di controllare che la sequenza di invocazione dei metodi sia compatibile con il comportamento previsto.

Le classi control possono essere collegate tra loro mediante associazioni: in tal caso, significa che una classe control usa le funzionalità di un'altra classe dello stesso tipo.

Le classi che usano questo stereotipo non possono avere attributi, ma solo una serie di metodi.

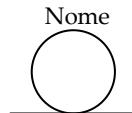
- ♦ **Lo stereotipo «entity»**

Lo stereotipo entità rappresenta un elemento della logica di accesso ai dati. I dati a cui si accede sono poi fisicamente rappresentati in una base di dati, o sul file system, o su repository distribuiti. Un oggetto entità corrisponde perciò a dei dati, oppure anche a delle risorse informative in genere. Il loro scopo è quindi quello di nascondere la reale struttura dei dati ai quali accedono.

Sono ammessi sia attributi, sia metodi. I metodi devono però consentire solamente la gestione dei dati, e non devono implementare la logica applicativa. In particolare, avremo:

1. *create()* e *remove()*, che consentono di creare o distruggere oggetti di quella classe;
2. I setter e i getter, per accedere in lettura e scrittura ai campi della classe (alcuni possono essere di sola lettura o di sola scrittura, ovviamente). È possibile omettere l'elenco dei getter e dei setter, sostituendoli semplicemente con il commento */* set e get */*.
3. Metodi statici di ricerca, come ad esempio *findAll()*, oppure *findByXXX()*.

Il simbolo sintetico per indicare un oggetto di tale classe è il seguente:



- ♦ ***Regole da seguire nella realizzazione del BCE***

Quando poi si realizza un BCE, si devono seguire alcune importanti regole:

1. I metodi delle classi boundary possono essere invocati solo dalle classi control o dagli attori;
2. Le classi boundary possono essere associate agli attori e/o alle classi control, ma mai alle classi entity; non possono neppure esserci legami tra classi boundary diverse. Se un boundary è associato ad un control, significa che quel control fornisce la logica applicativa necessaria allo svolgimento delle funzionalità associate al boundary.
3. Le classi entity possono essere associate solo alle classi control o ad altre classi entity. I metodi delle classi control non possono invece essere invocati tramite boundary, oppure direttamente dagli attori.
4. È il control ad invocare i metodi di entity, e mai viceversa.
5. Se una entity è collegata ad un'altra, significa che si ha una relazione tra i dati ai quali tali entity accedono. In particolare, si può avere un collegamento unidirezionale o bidirezionale. Se il collegamento è unidirezionale dalla entity Sorgente alla entity Destinazione, nella classe Sorgente dovrà essere presente il metodo `getDestinazione()` oppure `getListaDestinazioni()`, a seconda della cardinalità, mentre nella classe destinazione non dovrà essere presente alcun getter per gli elementi di Sorgente. Se invece il collegamento è bidirezionale, in Destinazione dovrà essere presente il metodo `getSorgente()` oppure `getListaSorgenti()`.

Si noti inoltre che se il collegamento è unidirezionale, da Sorgente è possibile invocare i metodi di Destinazione, ma non viceversa.

6. Come suggerimento pratico per procedere, è consigliabile:

- a) Occuparsi per prima cosa dei boundary, inserendone uno per ogni use case, e tenendo conto che un boundary è il raggruppamento di un insieme di screen presenti nello UX.
- b) Passare poi agli entity.
- c) Ci si occupa infine dei control (solitamente inserendone uno per ogni control).

7. Nella progettazione delle classi entity si deve tener conto che tali classi non devono rappresentare lo schema ER dei dati, ma devono essere solo uno schema per l'accesso ai dati. Bisogna però fare alcune considerazioni simili a quelle che si fanno solitamente nella progettazione delle basi di dati: ad esempio, i campi calcolabili non vengono indicati, ma si forniscono dei metodi che consentano di calcolarli.
8. In una entity devono essere indicati tutti i metodi che sono necessari a fornire le funzionalità richieste. È poi possibile aggiungerne altri, qualora risulti facilmente intuibile che tali metodi si possano rendere utili.
9. Se si ha un'associazione tra due classi control, significa che si ha una dipendenza tra tali classi per la realizzazione della logica applicativa. Ciò corrisponde alla classica suddivisione dei ruoli, tipica del modello *divide et impera*.

- ♦ ***Gestione delle sessioni***

In un diagramma logico, occorre anche rappresentare le sessioni. Per farlo, è possibile definire un passaggio di parametri tra le varie pagine, oppure, più spesso, si usa l'oggetto predefinito `HttpServletRequest`: due server page che condividono la stessa sessione devono avere una associazione alla stessa `HttpServletRequest`. Anche il controller può accedere alla classe `HttpServletRequest`.

- ♦ ***L'oggetto HttpServletRequest***

Esiste poi un'altra classe predefinita, detta `HttpServletRequest`: questa classe viene usata dalle server page che gestiscono dei link con parametri o dati provenienti dalle form. Le pagine client invece non devono essere collegate con la classe `HttpServletRequest`, perché di fatto è il browser ad occuparsi del passaggio dei dati, e non la pagina client (o la form) stessa.

- ♦ ***Diagrammi di sequenza***

Anche nel caso dei diagrammi di analisi può essere necessario definire la logica di interazione, indicando le possibili sequenze dei messaggi tramite diagrammi di sequenza. Realizzare un diagramma di sequenza in questa fase serve sia per evidenziare la dinamica del sistema, sia per verificare che per ogni funzionalità richiesta dall'utente attraverso l'invocazione di un metodo di una boundary, sia presente una serie di chiamate in grado di svolgere tale compito.

5. La fase di progettazione logica (o design)

In che cosa consiste la fase di progettazione logica?

La fase di progettazione logica è quella nella quale si rende necessario effettuare le scelte progettuali (si parla per la prima volta di pagine Web) e si delinea definitivamente la struttura che il WIS dovrà avere. In particolare, si distinguono 2 diverse fasi:

1. *La vista logica*

In questa fase si definiscono quali funzionalità debbano essere realizzate lato server e quali lato client.

2. *La vista dei componenti*

In questa fase invece, a partire dalla vista logica, si individuano i componenti software che compongono il sistema finale.

La vista logica (logical view)

La vista logica viene realizzata a partire dallo schema UX e dallo schema BCE: lo schema di vista logica consentirà infatti di “mettere insieme” tali schemi, realizzandone uno che ha un livello di dettaglio maggiore. Il risultato che si ottiene al termine di questa fase è uno schema UML arricchito con opportuni stereotipi. Gli stereotipi usati sono, oltre a control ed entity (che vengono “ereditati” dal BCE):

- ♦ ***Lo stereotipo «client page»***

Una client page è una pagina così come essa viene vista dal lato del client. Naturalmente, la client page può esser di tipo thin o fo tipo fat. Solitamente queste pagine non hanno né metodi, né attributi (eventuali metodi e attributi corrispondono alle funzioni di script e ai dati da esse usati, ma solitamente li ometteremo). Una client page può essere effettivamente una pagina statica, oppure il risultato dell’elaborazione di una server page (in tal caso, la client page non esiste da un punto di vista fisico).

- ♦ ***Lo stereotipo «server page»***

Una server page è una pagina Web che ha dei contenuti costruiti sul server in maniera dinamica, in risposta alle varie richieste che gli arrivano. Viene eseguita dal lato del server e interagisce con la logica di business e con la logica di accesso ai dati, o eventualmente anche con altri sistemi.

- ♦ ***Lo stereotipo «HTML form»***

Rappresenta i campi di input che sono parte di una client page. Gli attributi di questa classe sono quindi i cambi di input di una form HTML, e anche il tipo corrisponde sempre ad uno dei tipi che si possono attribuire mediante i tag di tipo input in HTML. Tra gli attributi deve essere incluso anche il pulsante submit. Una HTML form deve sempre essere legata ad una client page mediante un’associazione di composizione (rombo pieno).

Oltre agli stereotipi per le classi, esistono in questi diagrammi anche degli stereotipi per le associazioni:

- ♦ ***Lo stereotipo «link»***

Indica il collegamento tra due client page, oppure tra una client page e una server page o un control. Questo link può anche essere accompagnato dall’elenco dei parametri passati con la richiesta HTTP: i nomi di tali parametri sono indicati tra parentesi graffe. Tuttavia, per brevità useremo solo i simboli {}, omettendo l’elenco dei nomi dei parametri.

- ♦ ***Lo stereotipo «build»***

Associa una server page alla client page che viene generata dalla prima. In sostanza, se una server page “fa il build” di una client page, significa che la client page è l’output HTML della server page.

- ♦ ***Lo stereotipo «submit»***

È l’associazione tra una HTML form e una server page o una control: indica che la server page o il control elaborerà una richiesta con i dati inseriti all’interno dell’HTML form.

- ♦ ***Lo stereotipo «include»***

Associazione tra pagine: la pagina inclusa è elaborata all’interno della pagina che la include.

- ♦ ***Lo stereotipo «forward»***

Indica che una server page o un controller delega l’elaborazione di una richiesta a un’altra server page.

- ♦ ***Lo stereotipo «redirect»***

Indica che il controllo passa da una server page o un control ad una client page.

- ♦ ***Lo stereotipo «object»***

Indica che all’interno di una client page è usata un’altra classe della vista logica (ad esempio una classe che rappresenta una applet o un altro componente eseguibile inserito nella pagina HTML).

Vediamo adesso quali sono le regole pratiche che si devono usare per realizzare la logical view:

1. Per prima cosa, si inseriscono le pagine associate alle schermate dello UX: ad ogni screen si associa una client page e, nel caso in cui la pagina contenga anche dei dati dinamici, oltre alla client page si avrà una server page, legata alla client page mediante un'associazione con lo stereotipo build.
2. Le classi del tipo input form diventano delle classi HTML form, e saranno legate mediante composizione alla client page corrispondente alla screen alla quale era legata, nello UX, la corrispondente input form.
3. Per ogni metodo della screen, si definisce un'associazione di tipo link o submit verso un'altra pagina o verso il controller. In particolare, se non sono richieste elaborazioni (ma solo la visualizzazione di una pagina statica), allora il link andrà verso una client page, altrimenti andrà verso una server page.
4. Se una screen aveva i metodi next e previous, tali metodi non coinvolgeranno il controller: si avrà solamente un'associazione link dalla client page alla relativa server page (un'associazione per ogni metodo). Tale link, naturalmente, dovrà anche passare dei dati, perciò avrà i simboli {}.
5. Se una submit o un link vanno verso il controller, dal controller dovrà partire anche un'associazione di tipo forward verso una server page, oppure, se si deve visualizzare una pagina statica (ad esempio, si deve mostrare una pagina di errore), si avrà un redirect verso una client page.

La vista dei componenti (component view)

La vista dei componenti è un arricchimento della logical view, nel quale si definiscono i componenti fisici che costituiranno il WIS. Per fare ciò si aggiungono alcuni stereotipi:

- ♦ ***Lo stereotipo «static page»***

Una static page è una pagina che può essere richiesta direttamente dal browser, senza la necessità di elaborazioni dal lato server.

Queste pagine hanno solitamente il formato HTML (esplicitamente indicato nel nome attribuito alla classe). Ad esse corrisponderà perciò un file con quel particolare nome e quell'estensione.

- ♦ ***Lo stereotipo «dynamic page»***

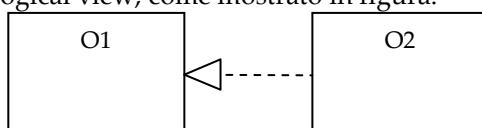
Si tratta di risorse che possono essere richieste dal browser; a seguito della richiesta, si effettua un'elaborazione lato server (direttamente oppure tramite forward della richiesta). Le dynamic page sono quindi tipicamente delle pagine .jsp (o, con altre tecnologie, pagine .php, ...); anche in questo caso, l'estensione viene indicata nel nome della classe.

- ♦ ***Lo stereotipo «package»***

Questo stereotipo non verrà usato. In ogni casi, indica la radice di file contenenti risorse che possono essere richieste al server.

Come si osserva da quanto detto, nella component view devono essere fatte precise scelte architettoniche: ad esempio, secondo le scelte fino ad ora seguite, nella component view si esplicita per la prima volta l'intenzione di usare la tecnologia Java, con pagine jsp,

Questo diagramma si ottiene dalla logical view, indicando quali static page o dynamic page realizzano i singoli elementi della logical view. Tale indicazione viene fornita associando la static page o la dynamic page a uno o più elementi della logical view, come mostrato in figura:



La figura precedente indica che O2 realizza O1, ovvero O2 è l'oggetto fisico (il file, in sostanza) che corrisponde all'entità logica O1.

1. In sostanza, per ogni server page si avrà una dynamic page, che sarà associata anche alle client page generate dalla server page cui è collegata, e a tutte le eventuali HTML form contenute nella client page.
2. Se invece una client page non è generata da un control o da una server page, allora tale client page sarà legata ad una static page. La static page implementerà anche le form contenute nella client page.
3. Al control sarà associata una classe dynamic page, la cui estensione però non sarà .jsp, ma tipicamente sarà .class (cioè si tratterà di una semplice classe Java).

6. Le fasi di realizzazione e deployment

Le ultime due fasi

Le fasi di realizzazione e di deployment sono quelle che chiudono il processo di sviluppo. Esse consistono nella realizzazione del codice relativo ai diagrammi fino a quel momento prodotti, e nell'installazione del software realizzato, sulle macchine client o server coinvolte.

Si realizzano così le pagine .html, le .jsp e le Servlet. Le pagine JSP possono anche essere sviluppate direttamente come Servlet. Oltre a questi elementi, bisognerà poi realizzare una serie di package di classi Java che realizzino la logica applicativa (potranno essere classi Java tradizionali, oppure JavaBean).

La suddivisione secondo il pattern MVC

Tutti questi elementi dovranno essere organizzati secondo il pattern MVC:

- ♦ ***Control***

Le classi che hanno lo stereotipo control corrispondono alla componente control del modello MVC, e possono essere realizzate come Servlet, classi Java o altri componenti software.

- ♦ ***Model***

Le classi con lo stereotipo entity rappresentano il model del pattern MVC e vengono realizzate come classi JavaBean, ad esempio.

- ♦ ***View***

Infine, la parte view è rappresentata dalle classi che hanno lo stereotipo static page oppure dynamic page, e vengono visualizzate come documenti html oppure jsp.

7. Esempio applicativo: WIS per un comune

Introduzione

Vogliamo ora affrontare, a scopo esemplificativo, un caso di studio concreto: quello relativo ad un comune di medie dimensione che intenda dotarsi di un sistema informativo per automatizzare la gestione dei propri atti amministrativi:

1. Delibere della giunta;
2. Delibere della consult;
3. Determine di servizio (emanate dai responsabili di servizio).

Fase di progettazione: analisi dell'azienda

La Pubblica Amministrazione Locale presa a riferimento è un comune di medie dimensioni (ca. 30.000 abitanti). All'interno vi lavorano 50 dipendenti suddivisi per AOO (Aree Organizzative Omogenee):

- Affari Generali
- Tributi
- Finanziaria
- Demografici
- Edilizia Pubblica
- Edilizia Privata
- Polizia Municipale

Gli uffici comunali sono suddivisi in 3 distaccamenti. Tutti i dipendenti ed amministratori hanno a disposizione un proprio terminale. Si ha inoltre una legislazione ad hoc che regolamenta la produzione e l'accesso agli atti amministrativi.

Al momento l'automatizzazione degli atti viene gestita attraverso un meccanismo di check-in check-out di documenti destrutturati (file Word). Ciò impone alcune problematiche:

- a. Si ha una forte limitazione nelle ricerche dei documenti attraverso il contenuto.
- b. Non esiste un sistema di monitoraggio dello stato di avanzamento di una pratica.

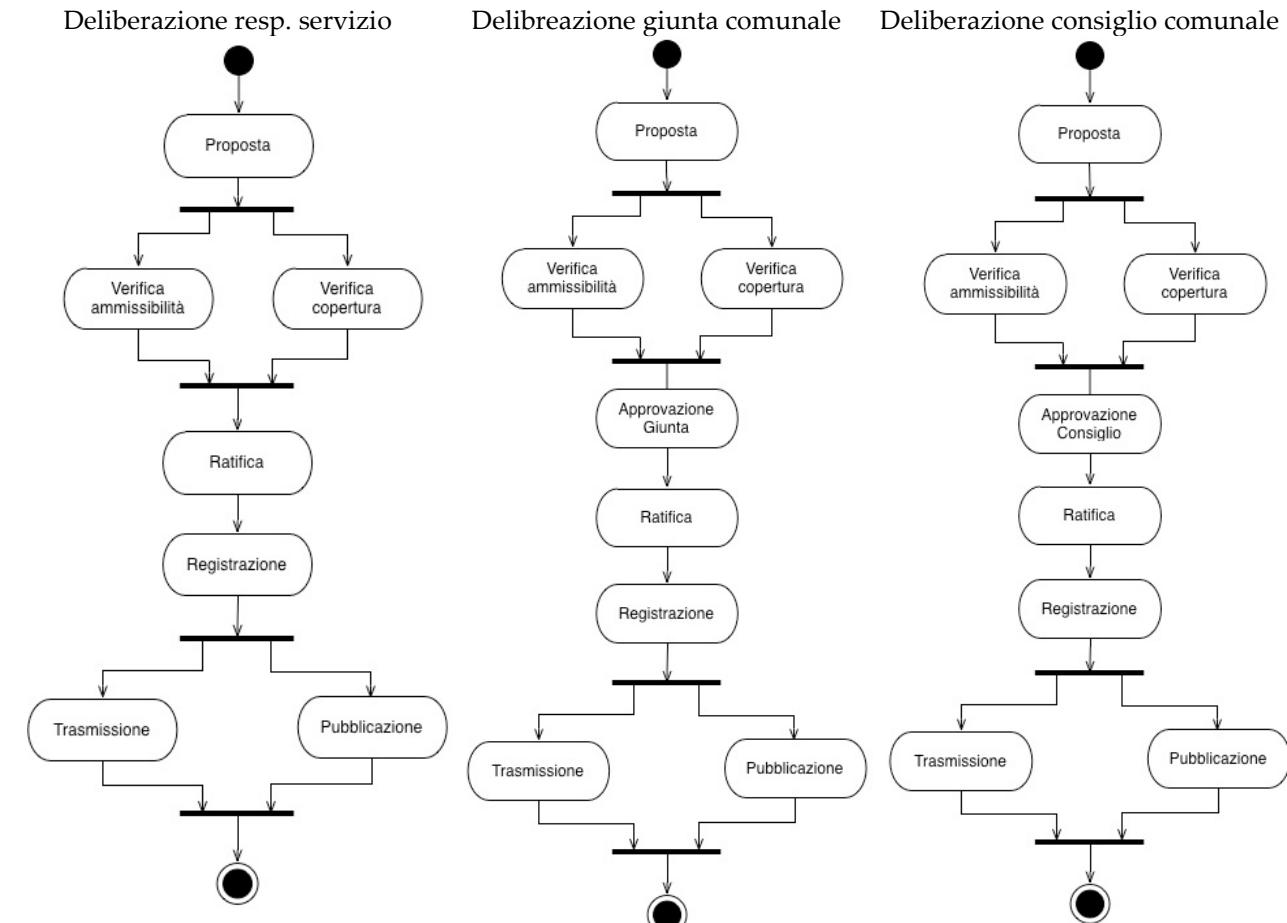
Tutte le sedi distaccate possiedono una LAN 100 Mb/s e sono interconnesse tramite CDN (2 Mb/s). Si hanno 2 server: il server con le applicazioni e un file server. Esiste poi un sito Web informativo pubblicato attraverso un contratto di hosting con una società esterna. Tutti i dipendenti e gli amministratori hanno a disposizione un proprio terminale ma la conoscenza informatica è mediamente bassa.

Fase di progettazione: analisi del dominio

Si possono identificare 3 macro processi:

1. Determinazione dei responsabili del servizio.
2. Deliberazione di Giunta Comunale.
3. Deliberazione di Consiglio Comunale.

Allo stato attuale, i 3 macro processi si svolgono come mostrato dai seguenti diagrammi delle attività:



Fase di progettazione: documento di Vision

Di seguito sono riportati alcuni stralci del documento di Vision:

- ♦ ***Introduzione***

Una amministrazione pubblica, indipendentemente dalle proprie dimensioni, presenta al suo interno un numero elevato di processi che, sotto il vincolo di esplicite normative, comportano una produzione e un flusso documentale rilevante.

Il flusso documentale in particolare obbliga l'amministrazione a impegnare diverse risorse ...

- ♦ ***Descrizione***

Le moderne tecnologie Scopo del progetto DELCOM è creare un sistema a supporto della gestione dei flussi documentali all'interno di una PAL di medie dimensioni quale è il Comune di Borgorosso. Referenti nella progettazione e realizzazione del sistema saranno sia i quadri interni alla PA che gli Amministratori.

- ♦ ***Background***

In particolare, il Comune di Borgorosso, presenta una serie di vincoli logistici ed organizzativi che rendono l'opera maggiormente difficoltosa ...

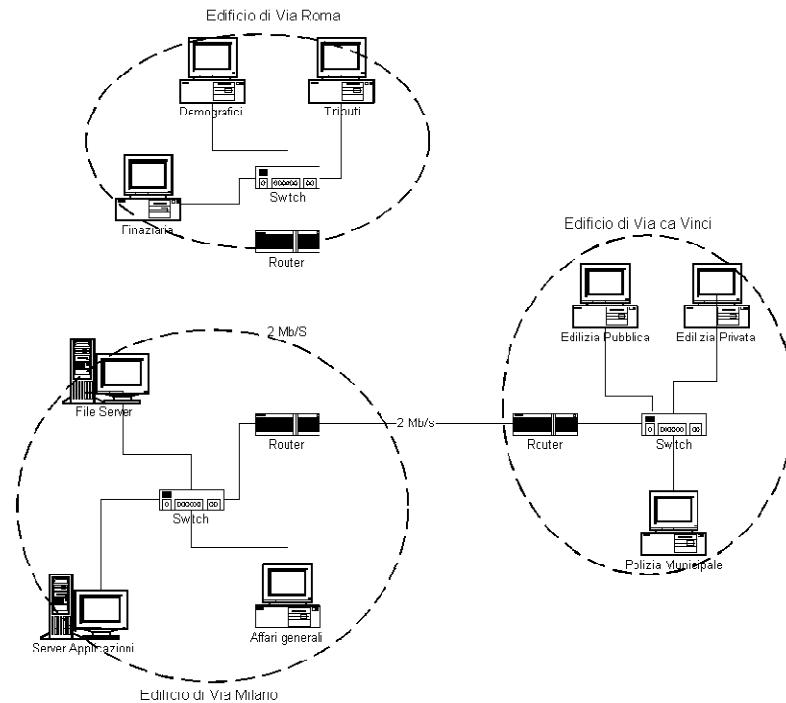
- ♦ ***Requisiti generali e funzionalità***

Attraverso DELCOM è possibile definire in modo flessibile il flusso documentale tale da seguire l'iter nel modo più aderente possibile le esigenze dell'Amministrazione. Ogni documento è accompagnato da un documento che attraverso i meta-dati contenuti permetterà una efficiente ed efficace base di conoscenza per una ricerca intelligente. Le tecnologie adottate permettono una forte integrazione con le basi dati esistenti tali da influire in modo positivo anche su flussi esterni.

Fase di progettazione: scelte architetturali

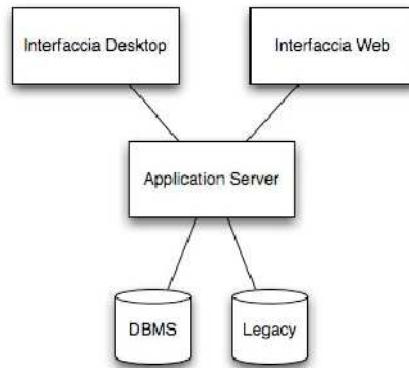
- ♦ **Rete**

La seguente figura rappresenta la struttura che avrà la rete:



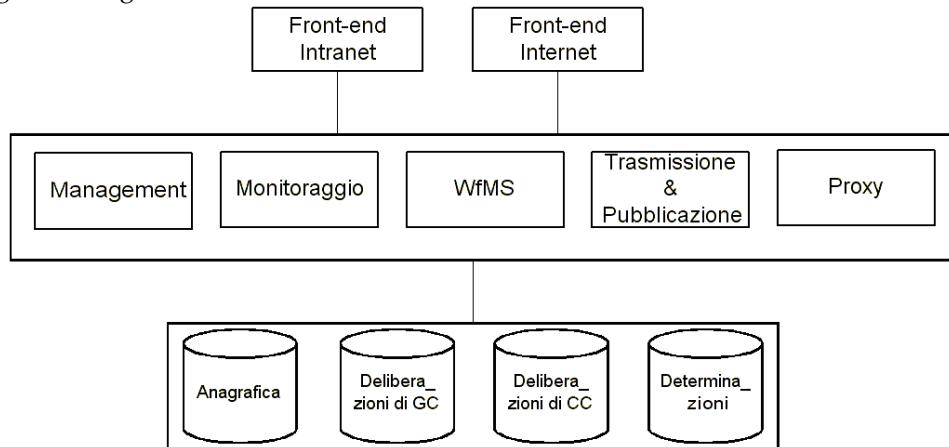
- ♦ **Sistema**

L'architettura generale del sistema è invece descritta dallo schema seguente:



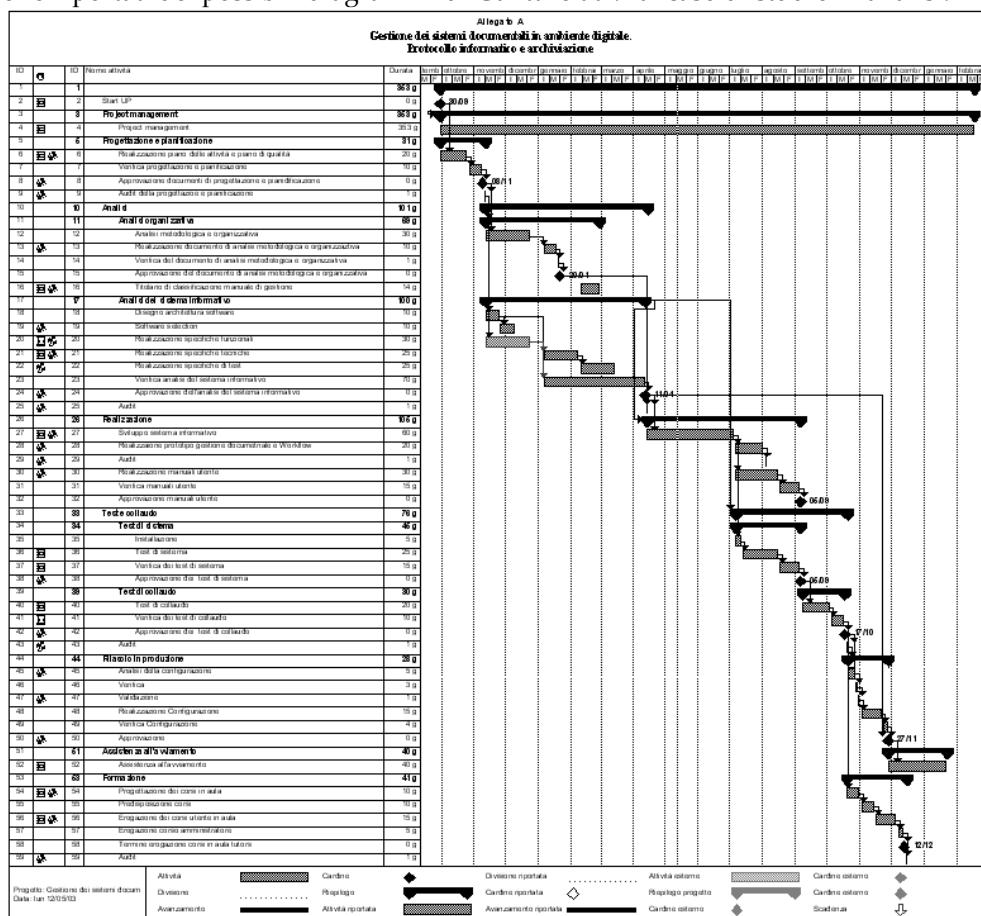
- ♦ **Funzionalità**

Infine, il seguente diagramma mette in mostra le funzionalità del sistema:



Fase di progettazione: pianificazione

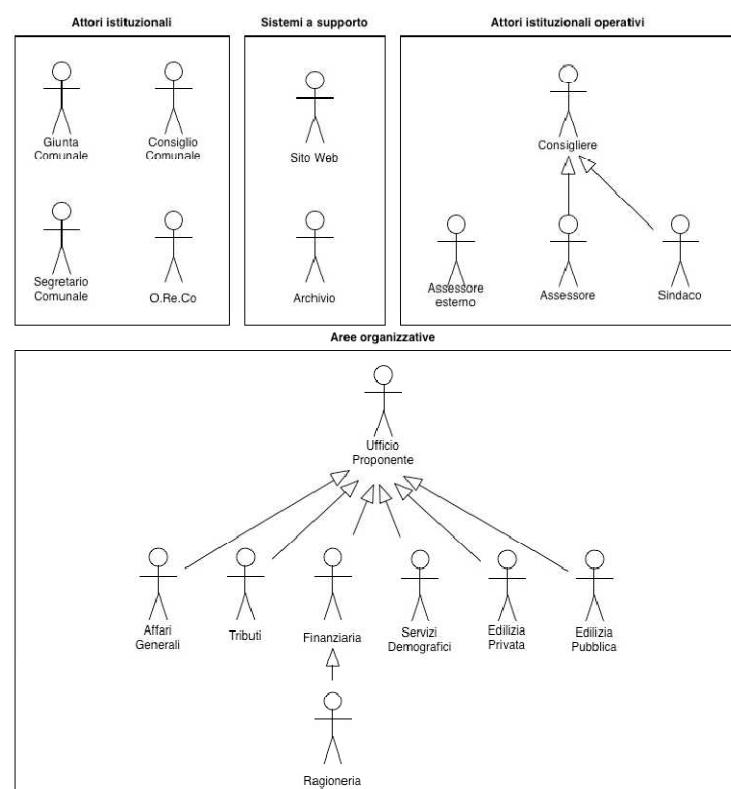
Di seguito sono riportati dei possibili diagrammi di Gantt relativi al caso di studio in analisi:



Analisi e specifica dei requisiti: identificazione degli attori

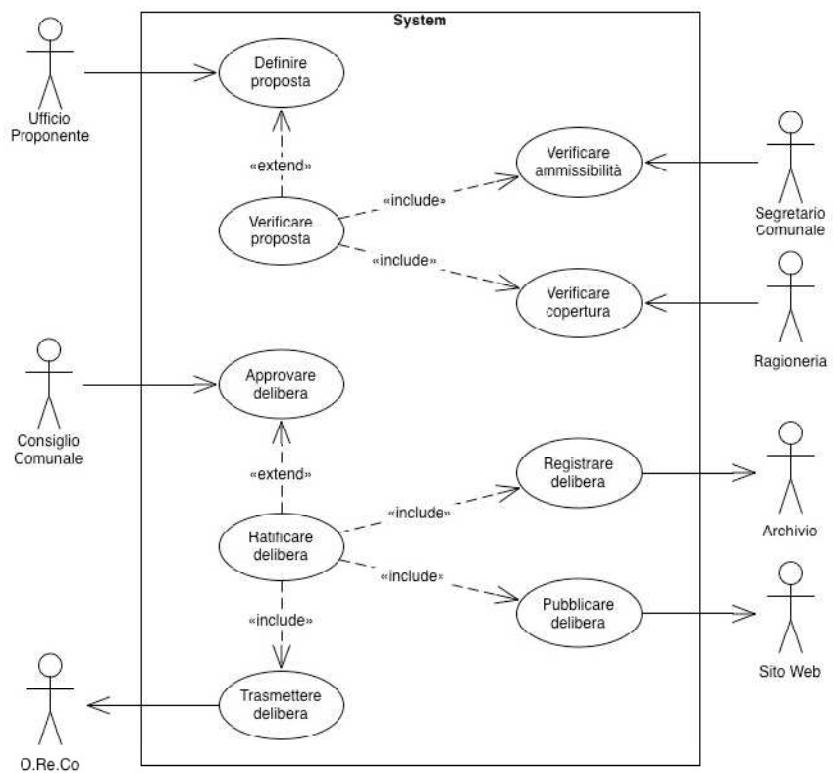
Gli attori che individuiamo nel nostro caso di studi sono mostrati nella figura a lato. Si noti che O.Re.Co. si occupa di approvare la legge da un punto di vista formale.

La figura mette inoltre in evidenza le gerarchie tra gli attori.



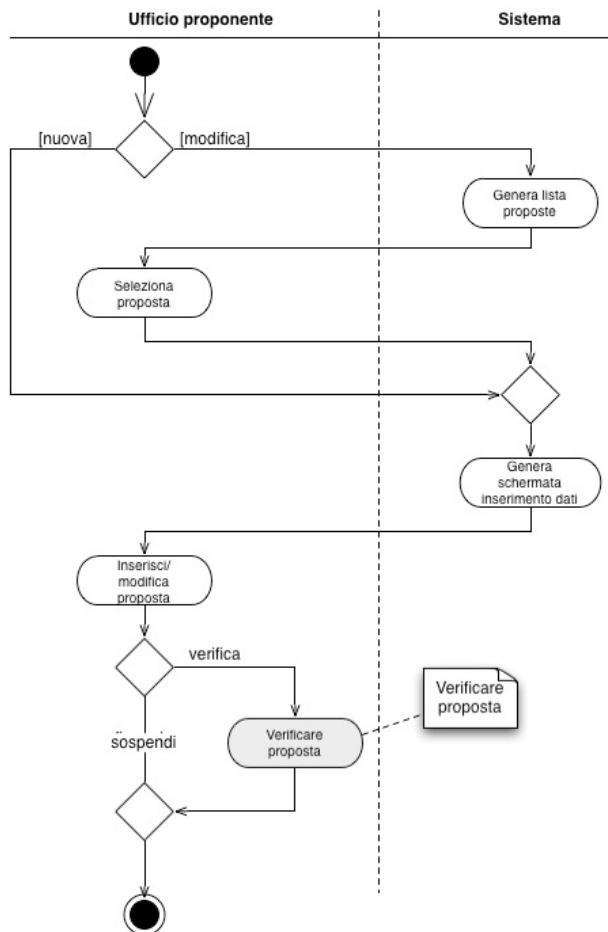
Analisi e specifica dei requisiti: diagramma dei casi d'uso

Il diagramma dei casi d'uso relativo alla situazione in analisi è il seguente:



Analisi e specifica dei requisiti: activity diagram

Possiamo ora tracciare l'activity diagram relativo allo use case di definizione della proposta:

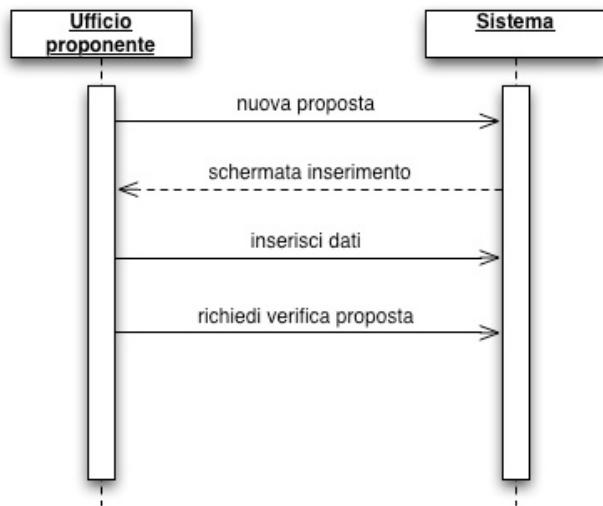


Analisi e specifica dei requisiti: sequence diagram e documentazione dei casi d'uso

A questo punto, dobbiamo documentare il caso d'uso in analisi. Sempre in riferimento alla definizione della proposta, avremo una documentazione testuale del tipo:

- Nome: Definizione proposta.
- Descrizione: permettere di definire le proposte pronte per essere approvate.
- Pre-condizioni: nessuna.
- Post-condizioni: la proposta deve essere inserita in un ordine del giorno.

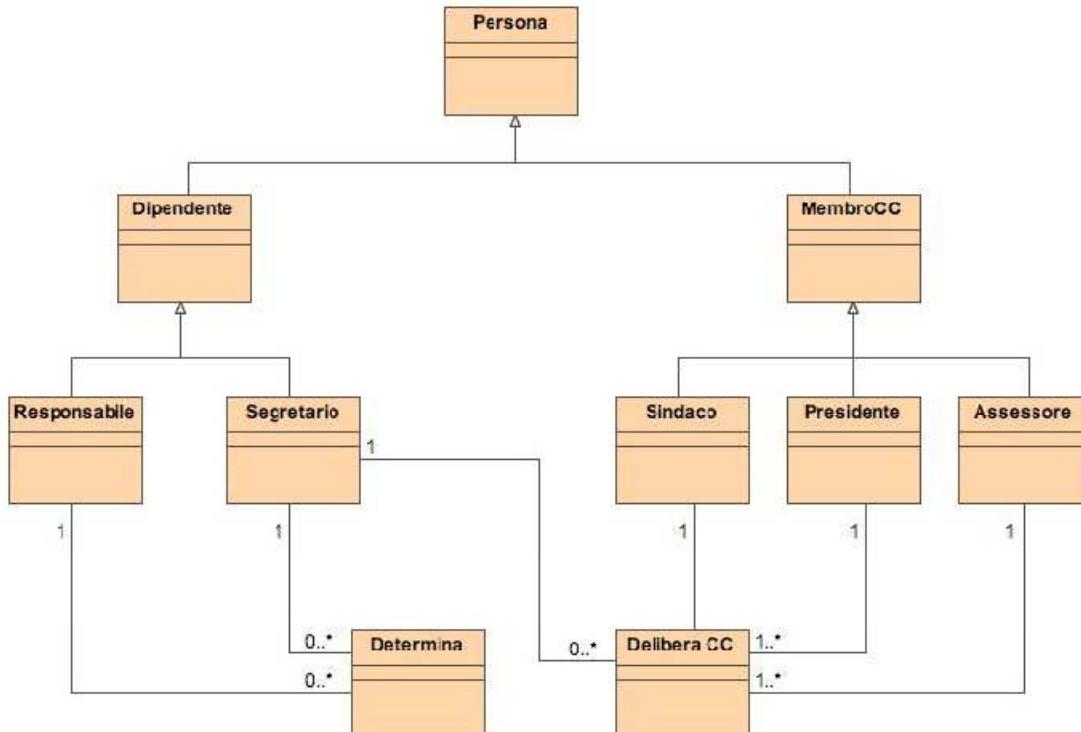
Dopodiché, possiamo rappresentare il sequence diagram relativo allo scenario di creazione di una nuova proposta (che è quello che si verifica più frequentemente):



Si noti che l'activity diagram, la documentazione e il relativo sequence diagram dovrebbero essere scritti anche per tutti gli altri casi d'uso (per brevità, qui ci siamo soffermati solo su uno dei casi d'uso presenti).

Analisi e specifica dei requisiti: definizione dei dati

Occorre ora realizzare un primo schema dei dati. Uno schema potrebbe ad esempio essere il seguente:

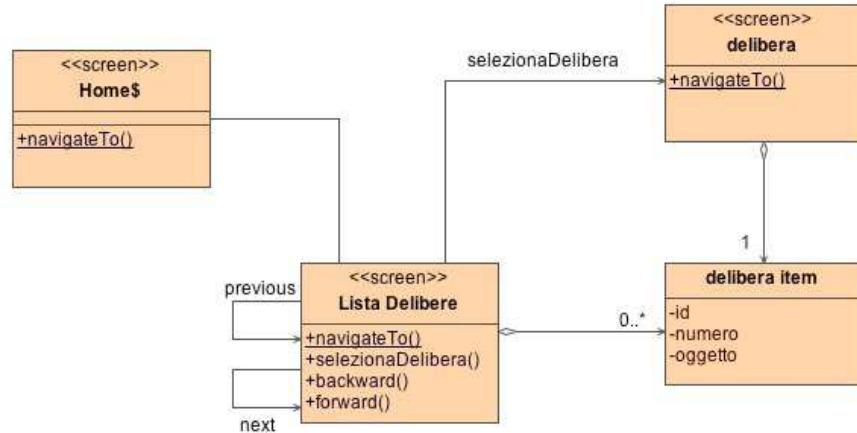


Analisi e specifica dei requisiti: UX

Supponiamo ora di voler definire lo schema UX. Di seguito sono mostrati alcuni esempi:

- ♦ **Elenco delle delibere**

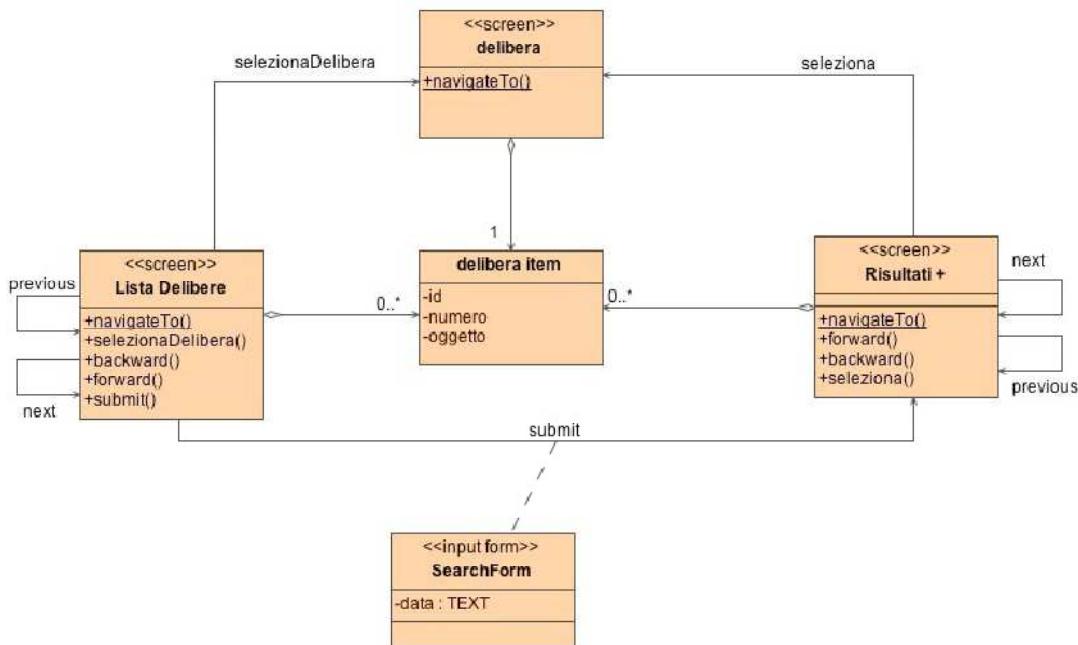
Il seguente schema mostra lo UX relativo ad un WIS che, a partire da una home page, permette di visualizzare una lista con tutte le delibere (spezzettata in una serie di pagine). Da tale lista, è poi possibile selezionare una specifica delibera per visualizzare solo i dati ad essa relativi.



Si noti che in questo modo la schermata delibera mostra solamente l'id, il numero e l'oggetto della delibera, ma non l'intero testo. Se avessimo voluto aggiungere tale dato, avremmo potuto inserirlo come attributo della screen.

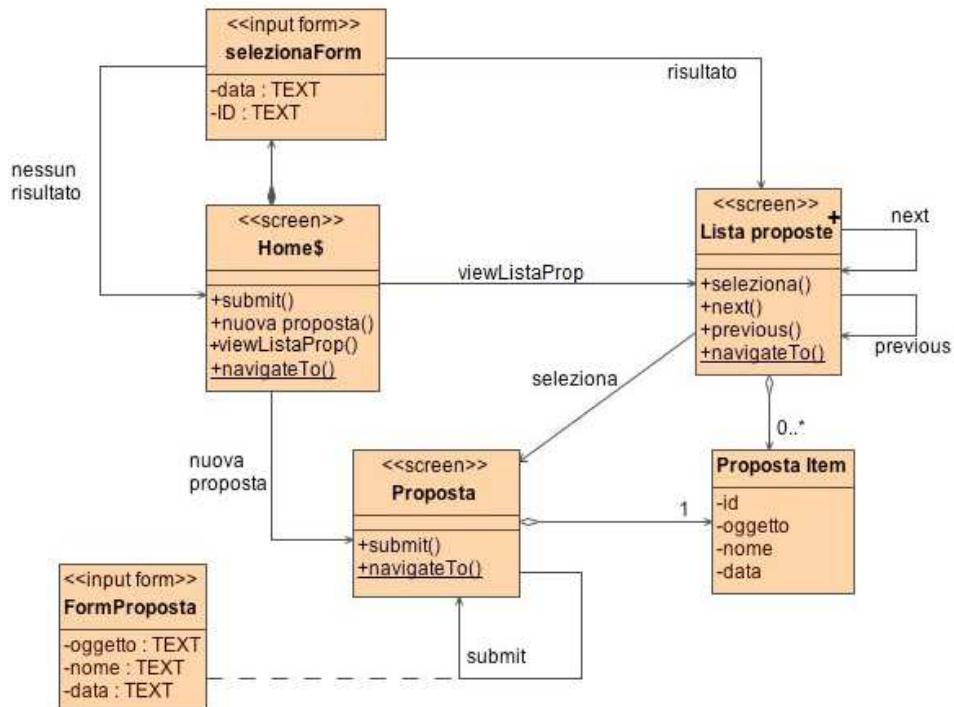
- ♦ **Ricerca di una delibera**

Se invece si mette a disposizione anche un servizio di ricerca della delibera, lo UX potrebbe essere:



- **Inserimento e modifica di una delibera**

A questo punto, possiamo ulteriormente modificare il precedente UX, in modo da consentire anche l'inserimento e la modifica di una delibera:



Si osserva quindi che è stata definita una home page (Home), dalla quale è possibile visualizzare la lista di tutte le proposte, oppure filtrare le proposte sulla base dei dati forniti all'interno di una form (selezioneForm). Dalla lista delle proposte, è poi possibile selezionare una delle proposte, così da visualizzare i suoi dati all'interno della schermata Proposta, che presenterà una form auto compilata (più l'id, che invece non è modificabile), all'interno della quale si possono modificare i dati. Premendo il tasto submit (ovvero confermando l'operazione), si visualizza nuovamente la stessa finestra (non sono previsti percorsi di errore).

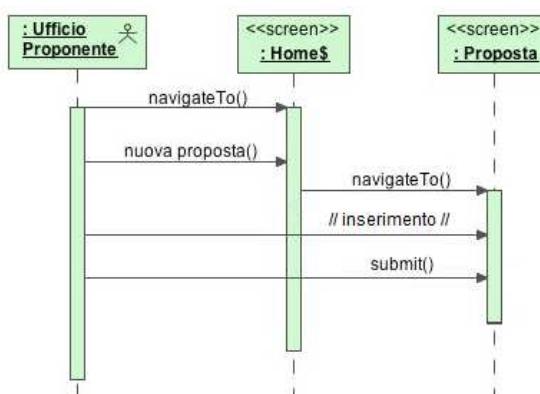
Inoltre, dalla home è possibile scegliere l'opzione per inserire una nuova proposta: la form visualizzata in questo caso è la stessa che si ha per la modifica.

Analisi e specifica dei requisiti: scenari di interazione

Per completare questa fase, a partire dall'UX sopra riportato, possiamo tracciare i seguenti diagrammi di sequenza:

- **Scenario di creazione nuova proposta**

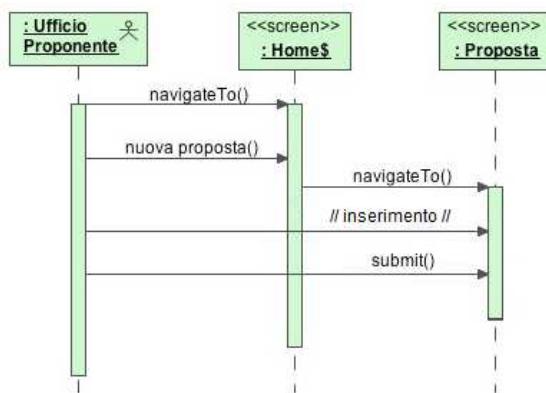
Lo scenario è il seguente:



Si osserva che, prima dell'evento `submit`, è stata inserita un'interazione tra "Ufficio proponente" e "Proposta" avente come etichetta il commento *inserimento dati*, per rappresentare le operazioni di digitazione dei dati da parte dell'utente.

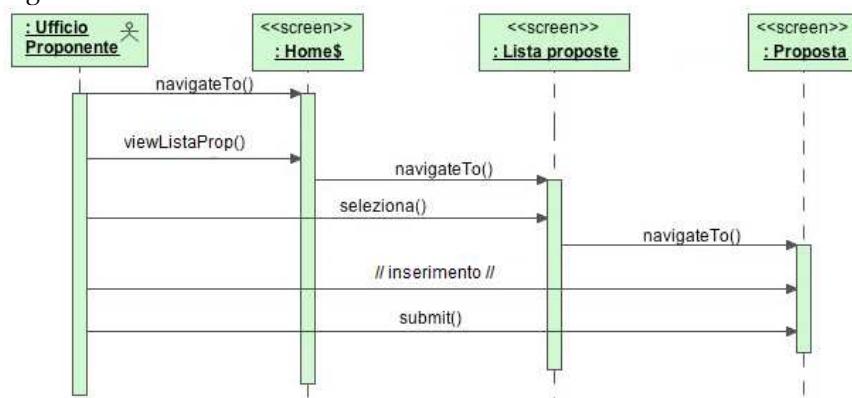
- ♦ *Scenario di creazione nuova proposta*

Lo scenario è il seguente:



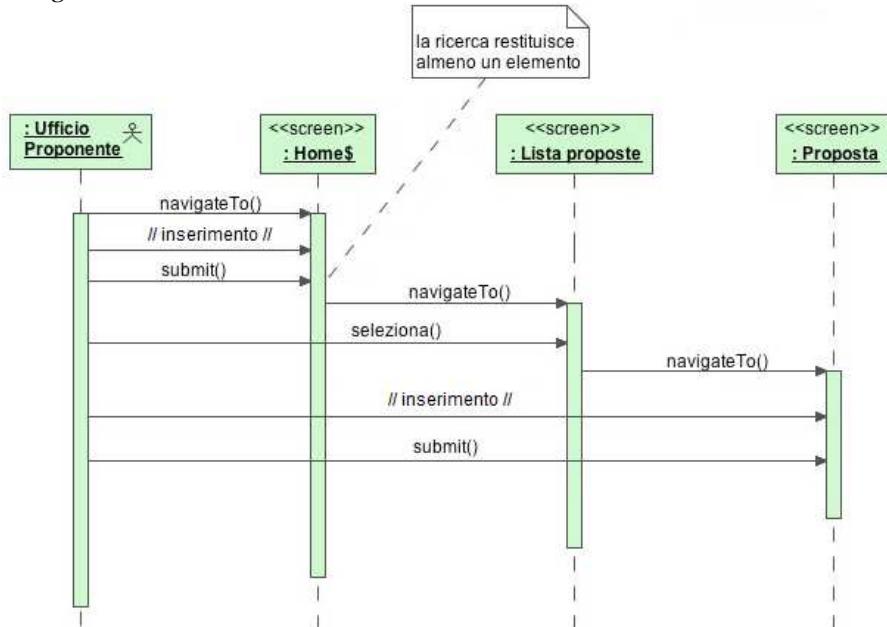
- ♦ *Scenario di modifica di una proposta selezionata dalla lista*

Lo scenario è il seguente:



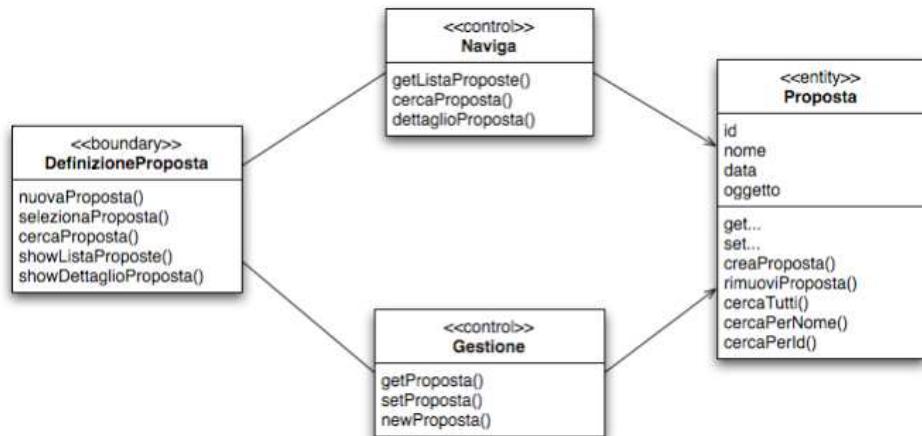
- ♦ *Scenario di modifica di una proposta con ricerca*

Lo scenario è il seguente:

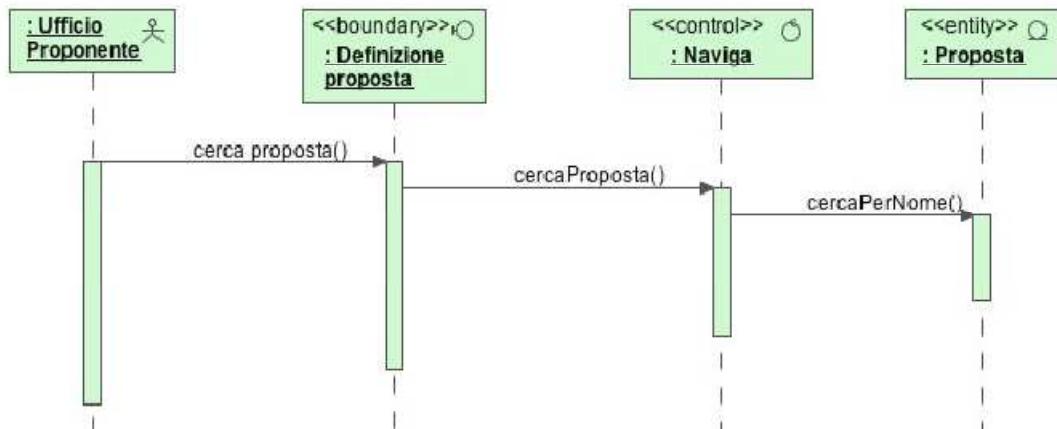


Fase di analisi dettagliata

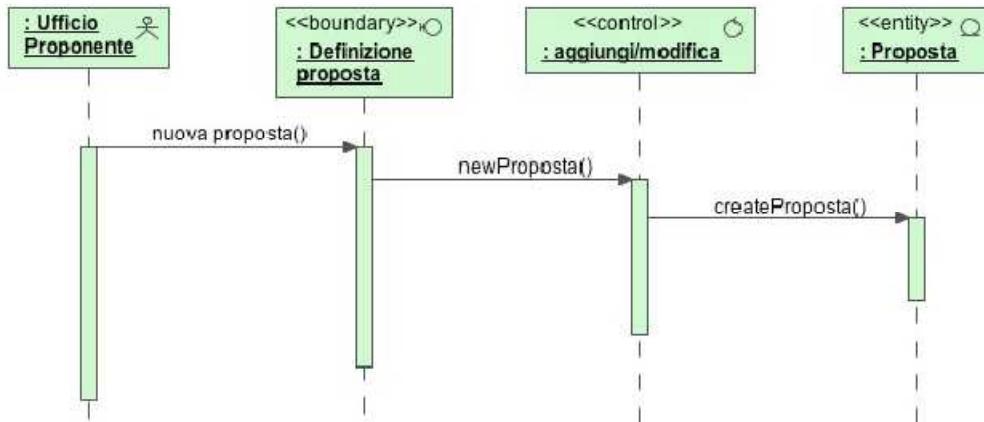
In fase di analisi dettagliata, possiamo supporre di utilizzare due control, uno per la navigazione uno per l'inserimento dati (si noti che in realtà avremmo potuto anche usare un unico control). Lo schema BCE è dunque:



Lo scenario relativo alla ricerca di una proposta viene invece così modificato:

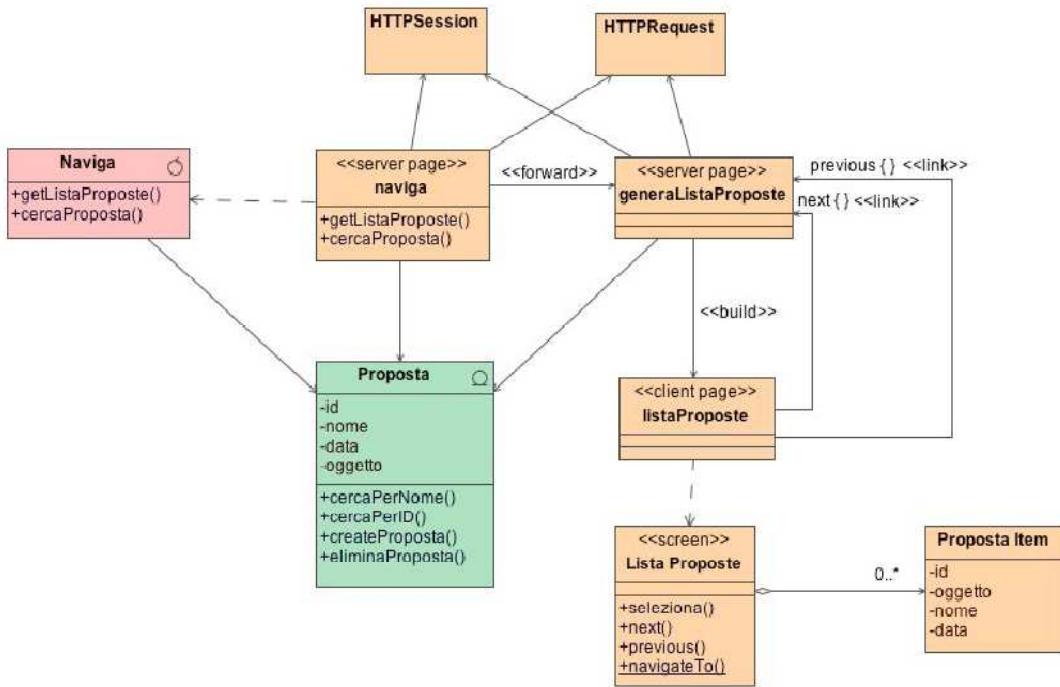


Mentre quello relativo all'inserimento di una nuova proposta sarà:



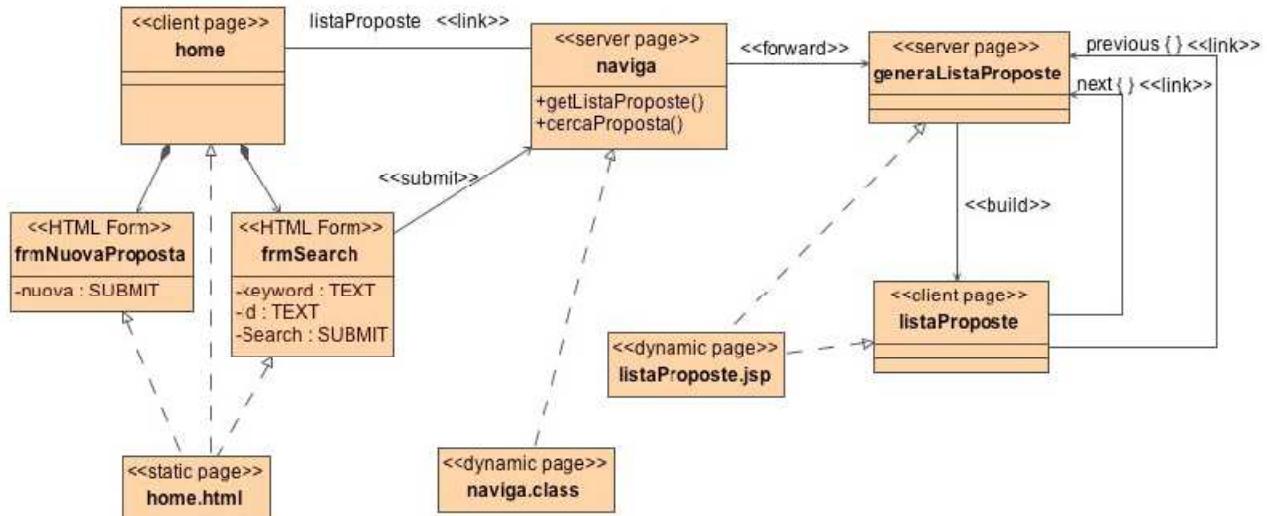
La fase di design: progettazione logica

Il risultato della fase di progettazione logica è riassunto nel diagramma seguente:



Progettazione dei componenti

Il risultato della progettazione dei componenti sarà invece:



Capitolo 11: Sicurezza dei WIS

1. Il problema della sicurezza

Definizione informale della sicurezza

La sicurezza nei sistemi informatici è l'insieme delle misure organizzative e tecnologiche tese ad assicurare ad ogni utente autorizzato (e a nessun altro) l'accesso a tutti e soli i servizi previsti per quel particolare utente, secondo le modalità e i tempi previsti.

Definizione ISO e requisiti di sicurezza

Formalmente, la sicurezza è l'insieme delle misure atte a proteggere i requisiti che si richiede che il sistema informativo soddisfi. Tali requisiti sono:

- ◆ **Integrità**
Il sistema deve impedire l'alterazione diretta o indiretta delle informazioni.
- ◆ **Riservatezza**
Nessun utente deve ottenere o dedurre dal sistema informazioni che non è autorizzato a conoscere.
- ◆ **Autenticità**
Le informazioni in transito e quelle memorizzate devono essere integre e contenere al loro interno anche elementi che permettano di identificarne il creatore, in modo da impedire il *ripudio* dell'invio o della ricezione di messaggi o informazioni.
- ◆ **Autenticazione**
Ogni agente (entità in grado di intraprendere azioni nel sistema) deve essere identificato prima di poter interagire con il sistema; anche il browser deve però autenticare il server con cui interagisce.

Tipi di minacce

Le minacce possono essere di diversi tipi:

- ◆ **Minacce fisiche**
Alcuni esempi sono: furti, danneggiamenti intenzionali, eventi accidentali (rottura impianto di condizionamento) o da disastri naturali (inondazioni, incendi, terremoti).
- ◆ **Minacce logiche**
Comprendono: sottrazione di dati (numeri di carta di credito memorizzati), creazione di punti di accesso nascosti al sistema (uso illegale delle risorse di calcolo e di comunicazione).
- ◆ **Accidentali**
Ad esempio: errori di configurazione, malfunzionamenti di programmi, errori di data entry.

Vulnerabilità e protezione

La sicurezza di un WIS deve essere considerata come un processo, e quindi bisogna tenerne conto in tutte le fasi dello sviluppo del sistema informativo stesso, cercando di sviluppare:

- ◆ **Protezione attiva**
Ovvero cercando di evitare gli attacchi alla sicurezza.
- ◆ **Protezione reattiva**
Ovvero cercando di rispondere efficacemente agli attacchi che vengono subiti, limitando i danni.
Se durante lo sviluppo del WIS non si tiene conto della sicurezza, il sistema avrà delle vulnerabilità, che verranno sfruttate per commettere violazioni della sicurezza. Per introdursi nel WIS, si userà poi un *exploit*, è un programma o una tecnica che, facendo leva sulle vulnerabilità del sistema, cerca di causare un comportamento anomalo ed imprevisto.

Politiche di sicurezza (policy)

Progettare la sicurezza significa definire delle regole che stabiliscano quali utenti possano accedere a quali dati del sistema e secondo quali modalità (sola lettura, lettura o modifica, cancellazione, ...). Un sistema di regole costituisce un *documento di policy* di sicurezza. È poi necessario definire i *meccanismi di sicurezza* in grado di implementare tali policy.

2. Attacchi nei sistemi distribuiti

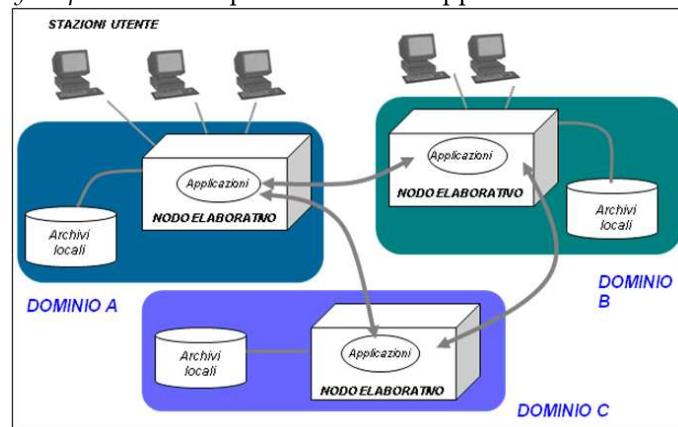
Architettura e problemi dei sistemi distribuiti

I sistemi informativi distribuiti hanno alcune importanti caratteristiche che introducono problemi nuovi e complessi riguardanti la sicurezza dei sistemi informativi stessi:

1. Rispondono alla necessità di condivisione e cooperazione su larga scala, perciò collegano sistemi di varie unità organizzative. Si ha così *condivisione* e *eterogeneità* di applicazioni, informazioni e risorse.
2. Si ha l'esigenza di preservare sistemi informativi e basi di dati preesistenti (legacy).
3. Questi SI evolvono sempre più verso soluzioni globali distribuite (*SI globali*).

Il concetto di dominio

Siccome sia le applicazioni, sia le basi di dati, saranno di tipo distribuito, allora è opportuno introdurre un concetto fondamentale: quello di dominio. Un dominio è l'insieme di tutte le risorse appartenenti ad uno specifico ente amministratore. Ogni dominio viene considerato come un'entità singola, connessa agli altri dominio mediante *gateway cooperativi* che esportano i servizi applicativi attraverso le *interfacce cooperative*.



Principali tipi di attacchi in architetture distribuite

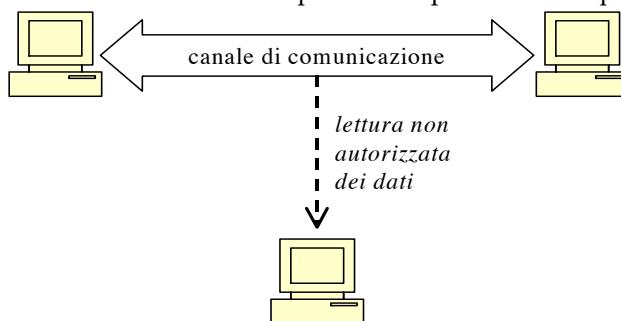
- ♦ ***Sniffing***

- Descrizione

Il termine *sniffing* viene usato per indicare un tipo di attacco che consiste nel catturare pacchetti di rete non destinati al nodo che effettua l'attacco. In sostanza quindi un nodo si mette in ascolto illegittimamente sul canale, leggendo così dati che non è autorizzato a leggere (ad esempio, potrebbe così intercettare delle password, o dati sensibili, ...). Si ha così una violazione della riservatezza.

- Individuazione e risoluzione

È difficile da individuare, perché la comunicazione tra i due terminali continua senza dare alcun segnale dell'attacco stesso. Il meccanismo di protezione più efficace è quindi la crittografia.



- ♦ ***Address spoofing***

- Descrizione

L'address spoofing è la generazione di pacchetti di rete contenenti l'indicazione di un falso mittente, finalizzata a nascondere la reale identità di chi invia i messaggi stessi. Si violano così le regole di autenticità e di autenticazione.

- Soluzione

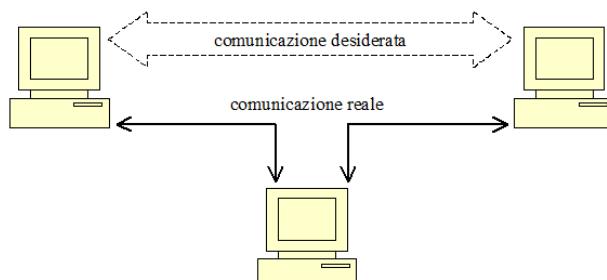
Una possibile soluzione a questo problema è l'adozione di meccanismi di autenticazione forte.

- ♦ ***Data spoofing***

Il data spoofing è una tecnica mediante la quale vengono creati dei dati falsi, oppure i dati scambiati tra due nodi vengono alterati. Viene così violata la proprietà di integrità.

- ♦ ***Hijacking***

Il termine *hijacking* indica una forma molto sofisticata di *data spoofing*, che consiste nel "dirottamento" di un canale virtuale. In questo modo, le informazioni vengono dirottate su un terzo nodo, che, prima di inviarle al vero destinatario, può leggerle ed eventualmente modificarle (si violano così riservatezza ed integrità).



- ♦ ***Denial-of-service (DoS)***

Questo attacco rende non disponibile un servizio, tenendolo impegnato in una serie di operazioni inutili o bloccandone completamente l'attività (DDoS). Ad esempio, questo attacco può essere effettuato "bombardando" il web server di richieste, in modo tale che sia occupato a rispondere alle richieste fittizie e non possa rispondere a quelle reali. Un possibile meccanismo per limitare questo tipo di problema è rappresentato dal filtraggio delle richieste.

- ♦ ***Phishing***

Il phishing è una tecnica usata per ottenere l'accesso a informazioni personali o riservate con la finalità del furto d'identità mediante l'utilizzo delle comunicazioni elettroniche, soprattutto messaggi di posta elettronica fasulli: grazie a messaggi che imitano la grafica e il logo dei siti istituzionali, l'utente è ingannato e portato a rivelare dati personali, come il proprio numero di conto corrente, il numero di carta di credito, codici di identificazione, ecc..., che poi vengono utilizzati illecitamente da chi ha effettuato l'attacco.

- ♦ ***Infezioni informatiche***

Appartengono a questa categoria:

1. *I trojan*

Sono programmi apparentemente utile, che nascondono porzioni di codice atte a realizzare azioni indesiderate o distruttive.

2. *I virus*

I virus sono dei malware in grado di infettare file in modo da riprodursi, facendo copia di sé stesso. Il virus mette poi in atto azioni indesiderate o distruttive (nella migliore delle ipotesi, si limita a provocare un pesante spreco di risorse).

3. *I worm*

Un worm è un programma che utilizza le tecniche di connessione in rete, ed i relativi protocolli, per trasferirsi da sistema a sistema. Anche un worm, come un virus, è in grado di replicarsi, ma non ha bisogno di legarsi ad un file eseguibile.

- ♦ ***Shadow Server***

Questa tecnica consiste nella creazione di un falso server in grado di sostituirsi a quello vero. In tal modo, verranno fornite informazioni sbagliate, oppure potranno essere catturati i dati introdotti dagli utenti. Tale attacco quindi comprende in sé gli attacchi di tipo sniffing, spoofing, DoS, e DNS Spoofing.

Possibili strategie di attacco

Alcune possibili strategie di attacco sono le seguenti:

- ◆ ***Uso di sistemi ponte***

Questa strategia consente nella diffusione di virus sulla rete. I sistemi così infettati vengono detti sistemi "ponte", e vengono a loro volta usati per attaccare altri sistemi, ottenendo così maggiore potenza di calcolo e maggiore banda.

Ad esempio, i sistemi ponte possono essere facilmente usati per la generazione di richieste, in modo da mettere in atto un attacco DoS (detto in tal caso *distribuite DoS*).

Per difendersi dall'uso di tali strategie, è opportuno agire in maniera preventiva, installando le patch, usando degli antivirus ed un firewall su tutte le macchine che costituiscono la rete, usando un sistema di protezione IDS ed effettuando il monitoraggio del traffico uscente.

- ◆ ***Sfruttamento di problemi di configurazione o bachi***

Alcuni attacchi sfruttano i problemi di configurazioni o le debolezze note dei sistemi (come i bachi software). Si rivela quindi particolarmente importante l'uso di patch (aggiornamenti software) per la risoluzione di problemi e bachi del software.

- ◆ ***Social engineering***

Anche se spesso sottovalutato, è questo il punto più debole di un sistema informativo: ad esempio, appartiene a questa categoria di tecniche di attacco la corruzione del personale autorizzato.

3. La crittografia

Introduzione: principali tecniche per il controllo della comunicazione

I principali meccanismi utilizzabili per proteggere la comunicazione tra un client ed un server sono:

1. La crittografia, ovvero la cifratura mediante chiavi dei dati scambiati.
2. I meccanismi di autenticazione, che può avvenire:
 - a. Mediante l'uso di password se i sistemi sono centralizzati;
 - b. Attraverso un sistema di distribuzione di chiavi (servizio di autenticazione).
3. Uso di un controllo dell'accesso, che può avvenire:
 - a. Sulle risorse (rese disponibili solo a chi è autorizzato).
 - b. Su file e su Web.
 - c. Su dati del DBMS.

La crittografia

- ◆ ***La crittografia***

Crittografare dei dati significa codificare l'informazione in modo da renderla intellegibile solo a chi possiede una (o più) *chiave* (o *chiavi*). Per eseguire la crittografia ci si avvale sempre di algoritmi noti e resi pubblici: la garanzia di confidenzialità è data solo dalla segretezza della chiave, che deve essere scelta tra un vastissimo numero di combinazioni e cambiata molto frequentemente.

- ◆ ***Classificazione degli algoritmi di crittografia***

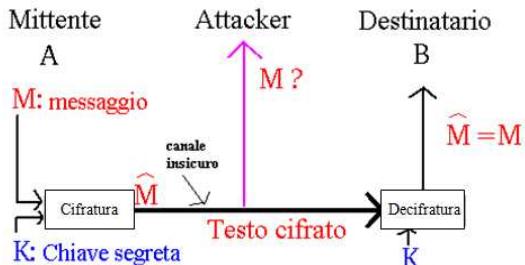
Gli algoritmi di crittografia si classificano in due categorie:

1. Algoritmi a chiave simmetrica
 - 1.1. Algoritmi di sostituzione.
 - 1.2. Algoritmi di trasposizione.
 - 1.3. Algoritmi di sostituzione e trasposizione (come lo standard DES).
2. Algoritmi a chiave asimmetrica (coppia chiave privata e chiave pubblica)
 - 2.1. RSA
 - 2.2. Diffie-Hellman

Crittografia simmetrica

La crittografia simmetrica consiste nella trasformazione di un messaggio mediante l'uso di una tecnica matematica (detta *chiave K*), che sia unica e condivisa. Il ricevitore, conoscendo K, sarà in grado di applicare l'algoritmo inverso e di tradurre il messaggio ricevuto nel messaggio originale.

Eventuali intrusi che dovessero intercettare il messaggio, se non in possesso della chiave, non possono decifrare il messaggio intercettato: si garantisce così la riservatezza della comunicazione.



La criticità di questo meccanismo è ovviamente rappresentata dalla fase di scambio della chiave, che deve avvenire fuori linea, oppure deve avvenire mediante tecniche più complesse, che prevedono che anche la chiave venga criptata.

Crittografia simmetrica: algoritmi di sostituzione

In questo caso, si hanno due possibili varianti:

- ♦ **Uso di un valore numerico come chiave**

La variante più semplice prevede che si usi come chiave un semplice numero intero K, e che si sostituisca ad una generica lettera dall'alfabeto quella che, nel tradizionale ordinamento alfabetico, si trova K posizioni più avanti. Ad esempio:

Se $K = 3$: ABCDEF diventa DEFGHI

- ♦ **Sostituzione monoalfabetica**

In alternativa, è possibile stabilire delle sostituzioni fisse ma più variabili tra loro: si crea cioè una relazione biunivoca e riflessiva tra l'insieme delle lettere dell'alfabeto. Ad esempio, si fissa la sostituzione:

Al posto di: A B C D E F G H I J H L M N O P Q R S T U V W X Y Z
Scrivere: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Crittografia simmetrica: algoritmi di trasposizione

La chiave è costituita da una parola di una certa lunghezza, ad esempio, la parola PROVATI. A questo punto, si analizzano le lettere che la costituiscono e si assegna a ciascuna di esse un numero, che rappresenta l'ordine che quella lettera assume se mettiamo in ordine alfabetico le lettere che costituiscono la parola stessa.

Ad esempio, riordinando le lettere di PROVATI, otterremmo AIOPRTV, perciò assegniamo ad A il valore 1, alla I il valore 2, e così via.... Quando poi dobbiamo trasferire un messaggio, lo scomponiamo in righe aventi un numero di caratteri pari alla lunghezza della chiave. Leggiamo poi il testo per colonne, secondo l'ordine imposto dai numeri attribuiti dalla chiave, e otteniamo il messaggio criptato da inviare. Nel nostro esempio, supponendo di dover trasmettere TRASFERIREUNAMILIONEAB:

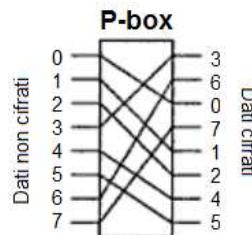
| | | | | | | |
|-------|---|---|---|---|---|---|
| P | R | O | V | A | T | I |
| 4 | 5 | 3 | 7 | 1 | 6 | 2 |
| <hr/> | | | | | | |
| T | R | A | S | F | E | R |
| I | R | E | U | N | M | I |
| L | I | O | N | E | A | B |

Di conseguenza, trasmetteremo: FNERIBAEOTILRIEMASUN.

Implementazioni in hardware

- ♦ **Implementazione di una S-box**

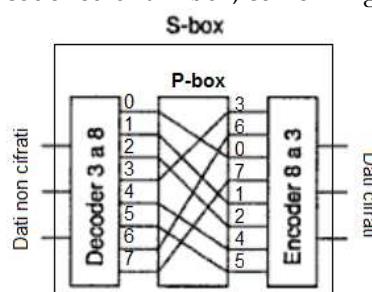
Implementare in hardware una P-box (permutation box), ovvero un dispositivo che applichi l'algoritmo di trasposizione, è in realtà molto semplice: basta cambiare l'ordine dei dati ricevuti, incrociando tra loro i collegamenti come mostrato in figura.



Ad esempio, se abbiamo in ingresso la stringa 01001101, in uscita avremo: 00011011.

- ♦ **Implementazione di una S-box**

Se invece vogliamo eseguire delle sostituzioni, dobbiamo implementare una S-box (substitution box), che è costituita da un decoder, un encoder ed una P-box, come in figura:

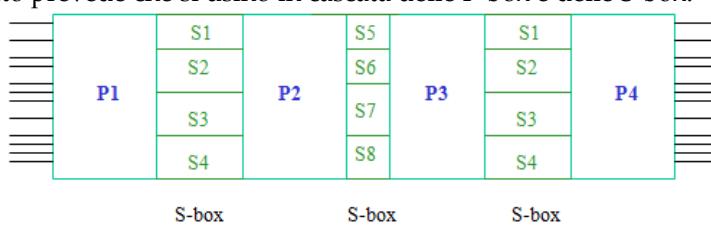


Il decoder pone a 1 l'uscita il cui numero corrisponde alla conversione in decimale del dato letto, mentre tutte le altre linee di ingresso della P-box sono poste a 0. La P-box quindi esegue una trasposizione. L'encoder riceve poi la sequenza trasposta e la converte in binario, mettendo così in uscita come dato cifrato il risultato di tale elaborazione.

Ad esempio, se forniamo il dato 010, il decoder attiverà la linea 2, quindi all'ingresso della P-box avremo (leggendo dall'alto), la sequenza 00100000. All'uscita della P-box avremo 00000100, e perciò il dato cifrato sarà 100 (corrispondente al numero decimale 4).

- ♦ **Cifratore per prodotto (product cipher)**

Il cifratore per prodotto prevede che si usino in cascata delle P-box e delle S-box:



In figura si è supposto di avere 12 linee in input. Si usano diversi livelli di P-box e di S-box. Ciascun livello di S-box però non è costituito effettivamente da una sola S-box, che richiederebbe $2^{12} = 4096$ fili incrociati), l'input è spezzato in 4 gruppi da 3 bit, ciascuno sostituito indipendentemente dagli altri. Inserendo un numero sufficientemente grande di stage nel product cipher, l'output può essere reso una funzione non lineare dell'input.

Lo standard DES

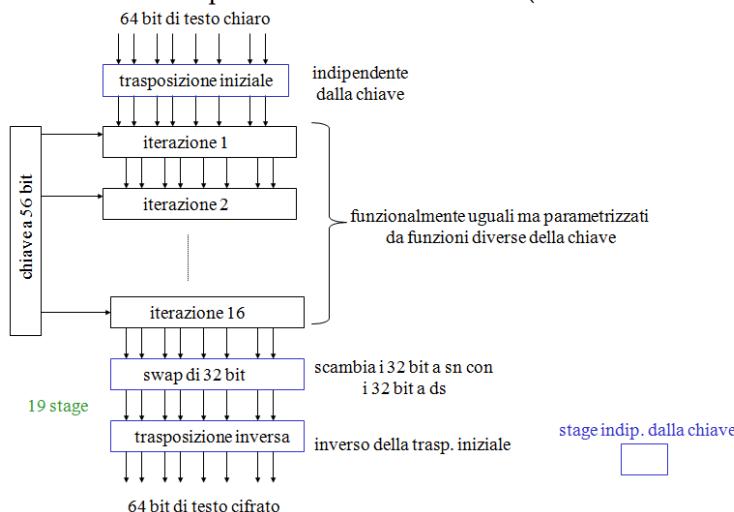
- ♦ **Che cos'è DES**

DES (Data Encryption Standard) è uno standard sviluppato negli anni '70 da IBM per la cifratura di informazione non classificata per conto del Governo USA. È diventato standard del NBS nel 1977, ma varie implementazioni hardware successivamente sviluppate lo hanno reso molto più veloce.

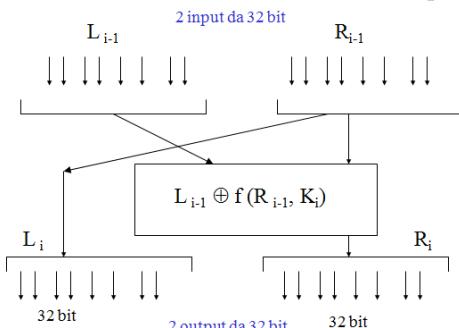
Lo standard prevede che venga eseguita la crittografia a blocchi da 64 bit di testo in chiaro.

- ♦ **Cosa prevede DES?**

Lo schema delle operazioni richieste da DES (divise in 19 diversi stage) è il seguente:



Ogni iterazione dipendente dalla chiave è del tipo:



Si ha quindi:

1. Uno stage iniziale di trasposizione, indipendente dalla chiave.
2. Una serie di 16 iterazioni che usano una chiave a 56 bit, in ciascuna delle quali l'input è diviso in 2 parti da 32 bit ciascuna: la parte di destra viene semplicemente riportata a sinistra, in output; la parte di sinistra invece viene elaborata prima di essere riportata in output a destra. L'elaborazione avviene usando una chiave K_i di 48 bit derivata dalla chiave a 56 bit. In particolare, tale elaborazione avviene nel modo seguente:
 - a) f espande i 32 bit R_{i-1} in un blocco di 48 bit e calcola l'EXOR bitweise con i 48 bit della chiave K_i .
 - b) Questo risultato intermedio viene diviso in 8 blocchi da 6 bit, che va in input a 8 S-boxes.
 - c) Ciascuna S-box converte il proprio input da 6 bit in un output da 4 bit.
 - d) Gli output delle S-boxes passano attraverso una P-box che esegue la permutazione sul proprio input da 32 bit e dà il risultato finale di f .
 - e) Al termine, si esegue l'EXOR con i bit di L_{i-1} .
3. Si scambiano i 32 bit a sinistra con i 32 bit a destra.
4. Si esegue la trasposizione inversa rispetto alla trasposizione iniziale.

- ♦ **Estensione a 56 bit**

Il DES può anche essere esteso con chiavi a 56 bit, mantenendo sempre lo stesso algoritmo. In particolare, è consigliabile usare il *triple-DES*, nel quale il procedimento viene iterato per 3 volte con 3 diverse chiavi a 56 bit.

Caratteristiche della crittografia simmetrica

- ♦ **Applicazioni commerciali**

Per decifrare i dati crittati con un meccanismo di crittografia simmetrica senza conoscere la chiave è necessario provare tutte le possibili chiavi, che risultano pari a 2^N (*brute force*). Maggiore è la lunghezza della chiave, maggiore è il livello di protezione offerto dall'algoritmo. Nelle applicazioni commerciali si usa $N \geq 100$ bit; nelle applicazioni ad alta sicurezza le chiavi hanno almeno 150 bit.

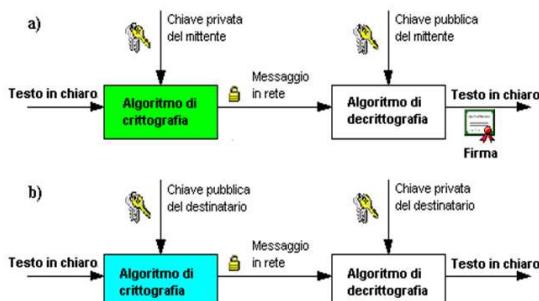
- ♦ **Vantaggi e svantaggi**

Il principale vantaggio della crittografia simmetrica è la velocità nell'esecuzione. Gli svantaggi sono:

1. Le parti per comunicare devono essersi incontrate
2. In caso di comunicazione uno a molti, diventa difficile tenere segreta la chiave.
3. Una volta che una chiave è compromessa, tutti i messaggi successivi sono intercettabili.

La crittografia asimmetrica

La crittografia asimmetrica ha un funzionamento che può essere descritto come mostrato in figura:



Il principio che viene applicato prevede che si usino: due diverse funzioni D ed E, che siano l'una l'inversa dell'altra, costruite in modo tale che D sia molto semplice, mentre E è una funzione molto complessa da calcolare. In particolare:

1. La funzione E (encrypt) usa una chiave K_e (*chiave pubblica*) e viene usata per crittare il messaggio.
 2. La funzione D (decrypt) usa una chiave K_d (*chiave privata*) e viene usata per de-crittare il messaggio.
- Siccome è molto difficile dedurre D da E, il livello di sicurezza è più alto. Naturalmente, deve valere:

$$D(E(P)) = E(P)$$

Algoritmo RSA

- ♦ *L'algoritmo*

Un algoritmo di crittografia asimmetrica è l'algoritmo RSA, basato sulla difficoltà di trovare i fattori di grandi numeri. In particolare, il procedimento è il seguente:

1. Si scelgono due numeri primi P e Q sufficientemente grandi (solitamente, maggiori di 10^{100}).
2. Si calcolano i numeri $N = P \cdot Q$ e $Z = (P - 1) \cdot (Q - 1)$.
3. Si sceglie un numero d primo rispetto a Z.
4. Si trova il numero e tale che $e \cdot d$ sia il più piccolo elemento della serie $i \cdot Z + 1$, $i \in \{0, 1, 2, \dots\}$ che sia divisibile per d.
5. Per crittografare, il messaggio è diviso in blocchi di 2^k bit, con $2^k < N$.
6. Per crittografare un blocco M di testo in chiaro, si applica la funzione:

$$E'(e, N, M) = M^e \bmod N$$
7. Per de-crittografare un blocco di testo cifrato c e ottenere il blocco in chiaro, si applica la funzione:

$$D'(d, N, c) = c^d \bmod N$$

- ♦ *Esempio*

Supponiamo ad esempio di voler trasmettere un blocco di dimensione 16 ($k = 5$), che indichiamo con $M = 14$ (dove 14 è la conversione in decimale della sequenza binaria che costituisce il blocco). Scegliamo come numeri $P = 13$ e $Q = 17$. Avremo perciò $N = 221$ e $Z = 192$. Di conseguenza, si può scegliere $d = 5$.

A questo punto, notiamo che la serie $i \cdot Z + 1$ inizia con gli elementi 1, 193, 385, 577... . Il primo di questi elementi divisibili per $d = 5$ è 385, perciò avremo $e = 385 / 5 = 77$.

A questo punto, possiamo crittare il messaggio:

$$E'(e, N, M) = E'(77, 221, 14) = 14^{77} \bmod 221 = 209$$

Infine, se proviamo a de-crittare il messaggio:

$$D'(d, N, c) = D'(5, 221, 209) = 209^5 \bmod 221 = 14$$

- ♦ *Osservazioni*

1. È dimostrato che si ha:

$$E'(D'(x)) = D'(E'(x)) = x, \quad \forall x: 0 \leq x \leq N$$

Diciamo quindi che E' e D' sono mutuamente inverse.

2. I parametri e e N possono essere visti come una chiave per la funzione di cifratura $K_e = \langle e, N \rangle$.
3. I parametri d e N possono essere visti come una chiave per la funzione di decifratura $K_d = \langle d, N \rangle$.

Protocollo Diffie-Hellman

- *Che cos'è il protocollo Diffie-Hellman?*

È un protocollo di agreement per le chiavi tra A e B che non condividono un segreto: su un canale insicuro A e B scambiano una chiave segreta (costruiscono cioè un segreto).

- *Descrizione dell'algoritmo*

In particolare, l'algoritmo prevede i seguenti passi:

1. Si sceglie un insieme di interi $I = [0, N - 1]$ (con N grande), e si sceglie un suo elemento E .
2. A sceglie casualmente un elemento a di I , calcola $E^a \bmod N$ e lo invia a B.
3. B sceglie a caso un elemento b di I , calcola $E^b \bmod N$ e lo invia ad A.
4. A questo punto, A calcola $K = (E^b)^a \bmod N$.
5. Infine, B calcola $K = (E^a)^b \bmod N$.

Vantaggi e svantaggi della crittografia asimmetrica

La crittografia asimmetrica ha lo svantaggio di avere una complessità che cresce in maniera non lineare rispetto alla lunghezza del blocco codificato. Alcuni vantaggi sono la possibilità di gestire le identità e la possibilità di avere la firma digitale.

La crittografia asimmetrica è indicata soprattutto per le reti poco strutturate.

4. Impronta e firma digitale

L'impronta

Anche se un intruso non dovesse essere in grado di decrittare le informazioni scambiate tra due nodi di rete, è possibile che tale nodo alteri i messaggi scambiati, rendendoli illeggibili. È allora importante disporre di opportune tecniche per evidenziare se un dato è stato manipolato e deve essere escluso dalle normali elaborazioni.

L'impronta dei dati (anche detta *digest*) può essere considerato un "riassunto" dei dati che si vuole proteggere da manipolazioni illegittime ed è generata tramite appositi algoritmi di "hash".

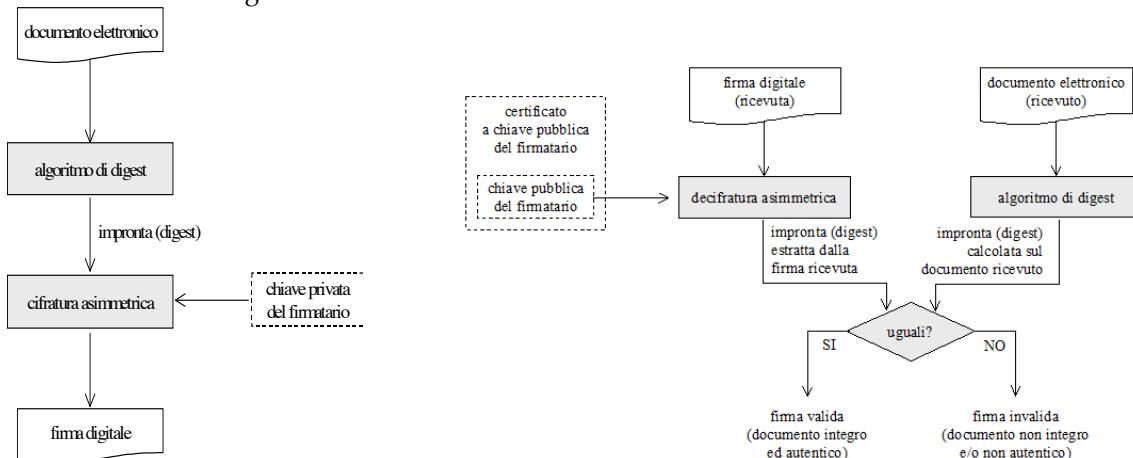
La firma digitale

La firma digitale è un meccanismo per garantire l'autenticità dei dati e identificare con certezza l'autore del messaggio. La firma digitale consente anche di dimostrare che i dati non sono stati modificati dopo essere stati firmati (è più forte della normale "firma autografa").

L'uso della firma digitale richiede un tempo di elaborazione relativamente lungo, perciò la firma digitale non viene normalmente usata in processi di rete che richiedono altre prestazioni, ma solo in soluzioni applicative in cui è direttamente coinvolto un agente umano.

La firma digitale dipende non solo dalla chiave privata del firmatario ma anche dai dati che sta firmando, quindi è diversa per ogni insieme di dati diverso che viene firmato: quindi, la firma digitale apposta su un documento elettronico non può essere copiata ed apposta su un diverso documento elettronico.

Gli schemi seguenti mostrano le operazioni che devono essere effettuate per l'invio e la ricezione di documenti con la firma digitale.



5. Generazione, conservazione e scambio di chiavi

Generazione di una chiave crittografica

La generazione delle chiavi deve essere fatta direttamente da chi effettuerà le operazioni crittografiche. Solo in casi eccezionali (ambienti fidati ad alta sicurezza) è lecito permettere che le chiavi vengano generate da un'entità diversa da quella che ne farà uso.

Inoltre, le chiavi crittografiche dovrebbero essere stringhe casuali di bit, mediante l'uso di un RNG (Random Number Generator); dalla qualità del RNG dipende la qualità del sistema di sicurezza.

Conservazione delle chiavi crittografiche

La conservazione delle chiavi è un problema importante: se le chiavi sono note agli attaccanti, tutte le altre misure di sicurezza possono essere aggirate. Si possono allora adottare diverse soluzioni:

1. Soluzione software: le chiavi vengono memorizzate in un file cifrato protetto da password.
2. Soluzione hardware: le chiavi vengono memorizzate all'interno di dispositivi hardware rimovibili:
 - a) Unità di memoria rimovibili (floppy-disk, CD-ROM).
 - b) Dispositivi "attivi", ossia in grado di memorizzare le chiavi e di svolgere autonomamente le operazioni crittografiche (Es: smart-card, token USB o PCMCIA, acceleratori crittografici).

Scambio di chiavi segrete

Le chiavi segrete possono essere scambiate in maniere diverse:

1. Con la tecnica OOB (Out-Of-Band), cioè la chiave viene distribuita tramite un canale diverso da quello su cui transitano i dati.
2. Se il destinatario dei dati cifrati dispone di una coppia di chiavi asimmetriche, il mittente può inviargli la chiave segreta cifrandola con la chiave pubblica del destinatario.

Scambio di chiavi pubbliche

- ♦ *Uso del certificato*

Nel caso in cui la chiave da scambiare sia di tipo pubblico non è importante la sua segretezza, l'unico problema è associarvi correttamente l'identità del possessore. Le chiavi pubbliche vengono diffuse tramite una struttura dati denominata "certificato a chiave pubblica" o PKC (Public-Key Certificate). Esistono vari formati di PKC ma quello attualmente più usato è il formato X.509v3.

- ♦ *Cos'è il certificato a chiave pubblica?*

Il certificato è una struttura di dati per legare in modo sicuro una chiave pubblica ad alcuni attributi (tipicamente il certificato lega la chiave a una persona). Il certificato ha una scadenza temporale ed è revocabile sia dall'utente che dall'emittitore. Inoltre, il certificato è firmato in modo elettronico dall'emittitore, l'autorità di certificazione (CA).

Gli attributi contenuti nel certificato sono:

- a) Identificatore di chi ha emesso il certificato
- b) Identificatore del soggetto che possiede la chiave privata corrispondente a quella pubblica.
- c) La chiave pubblica del soggetto e le funzionalità per le quali è stata certificata, che possono essere:
 1. funzionalità di basso livello: es. non ripudio, scambio di chiavi segrete
 2. funzionalità applicative: es. protezione della posta elettronica
- d) Altri attributi: la versione del protocollo, il numero di serie del certificato, il periodo di validità, ...

- ♦ *Firma digitale della CA*

Per garantire che il certificato non possa essere alterato, esso viene protetto con la firma digitale dell'autorità di certificazione che lo ha emesso. Un'autorità di certificazione o CA (Certification Authority) è un organismo creato per assolvere compiti organizzativi (identificare i soggetti che richiedono un certificato) e tecnici (creazione dei certificati).

La CA ha anche funzionalità accessorie come la distribuzione dei certificati o loro revoca.

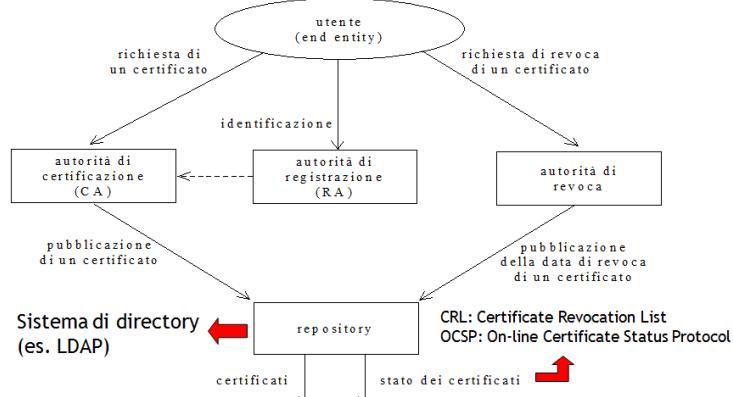
- ♦ **La PKI**

Quando si usano chiavi asimmetriche, è opportuno dotarsi di un'Infrastruttura a supporto delle chiavi stesse, detta *infrastruttura a chiave pubblica* o PKI (*Public-Key Infrastructure*).

La PKI è un'infrastruttura che fornisce le seguenti funzioni base ad comunità di utenti:

- Emissione dei certificati* a chiave pubblica, dopo aver espletato i dovuti controlli tecnici e procedurali.
- Revoca* dei certificati a chiave pubblica (es. furto della chiave privata associata alla chiave pubblica).
- Distribuzione* dei certificati a chiave pubblica e delle informazioni circa i certificati revocati.

Opzionalmente, la PKI può fornire anche supporto per la validazione di una firma digitale tramite funzioni di identificazione dell'ora a cui è stata apposta la firma ("marca temporale"), del ruolo ricoperto dall'individuo che ha firmato ("certificazione di ruolo") e del motivo per cui è stata apposta la firma ("politica di firma").



Quando si riceve una chiave pubblica attraverso un certificato, occorre controllare l'integrità e validità della firma: il digest calcolato sul certificato (ossia sulla chiave pubblica e sui dati ad essa associati) viene confrontato col digest estratto dalla firma digitale apposta dalla CA che ha emesso il certificato. Se questo confronto ha esito positivo, il certificato è sicuramente integro ma resta da verificare se la CA che lo ha emesso è una CA fidata. Per risolvere questo problema, si possono adottare le seguenti soluzioni:

1. Viene mantenuto un elenco delle chiavi pubbliche delle CA fidate.
2. Si creano delle gerarchie di certificazione, in cui una CA emette certificati non solo per utenti ma anche per altre CA subordinate che hanno il compito di certificare insiemi diversi di utenti.

- ♦ **Standard di PKI**

Non c'è ancora una PKI (Public Key Infrastructure) universale: vari standard si contendono ancora oggi tale ruolo. Tra i più importanti, ricordiamo X.509v1 (ISO), X.509v3 (ISO + IETF) e PKCS (RSA, in parte compatibile con X.509v3).

Autenticazione degli agenti

Prima di permettere agli agenti di aprire un canale di comunicazione o di accedere a delle informazioni è necessario che venga effettuata la loro autenticazione. Una persona può essere identificata sfruttando una o più delle seguenti caratteristiche:

- a) Un'informazione che essa conosce (es. password) (*something you know*).
- b) Un oggetto che essa possiede (es. una carta magnetica) (*something you have*).
- c) Una sua caratteristica fisica (es. impronta digitale, retina, DNA) (*something you are*).

6. Architetture di sicurezza

Le architetture di sicurezza

Le tecniche discusse finora permettono di realizzare una serie di funzionalità di sicurezza che possono essere inglobate in diversi elementi dell'architettura di un sistema informativo. Progettare un'architettura di sicurezza significa trovare la combinazione migliore di elementi che permetta di raggiungere il livello di sicurezza globale desiderato.

Praticamente a tutti i livelli di rete del modello OSI è possibile realizzare misure di protezione. Più basso è il livello a cui si opera e più la misura di sicurezza sarà generica ed aspecifica.

È quindi importante stabilire dove posizionare i meccanismi di protezione del sistema. Le possibilità sono:

1. Al livello dei protocolli di rete:

In tal modo si ottiene automaticamente la protezione di tutti i dati trasportati dal protocollo, ma risulta molto difficile distinguere tra i dati generati da applicazioni diverse o da utenti diversi.

2. Al livello applicativo

Occorre in tal caso modificare le applicazioni esistenti.

Le architetture di sicurezza possono prevedere:

1. L'uso del firewall, un sistema che protegge una rete da attacchi provenienti da altre reti.
2. La creazione di una rete privata virtuale (VPN), che collega in modo sicuro singoli nodi o intere reti attraverso canali di comunicazione non sicuri.
3. Misure di sicurezza a livello applicativo.
4. Sistemi per la protezione della configurazione SW dei sistemi di elaborazione e degli apparati di rete.
5. Sistemi automatici di verifica della sicurezza e di monitoraggio in tempo reale.

Il firewall

♦ *Cos'è il firewall?*

Il firewall è una difesa di tipo perimetrale: essi infatti verifica che i pacchetti destinati alla rete protetta dal firewall stesso siano in linea con la politica di sicurezza aziendale e non possano causare danni ai sistemi interni. Il firewall viene normalmente posizionato all'interno del gateway che collega una rete con le realtà esterne. Solitamente, il firewall è un sistema costituito da varie componenti sia hardware che software. Talvolta può essere realizzato tramite un unico componente hardware che ospita diversi processi software, ma in generale è preferibile che sia costituito da più elementi distinti che costituiscano linee di difesa multiple.

♦ *Le tre leggi del firewall*

1. Il firewall deve essere l'unico punto di contatto tra la rete interna e l'esterno.
2. Solo i pacchetti "autorizzati" possono attraversare il firewall.
3. Il firewall deve essere un sistema "sicuro" esso stesso.

♦ *Architettura del firewall*

Il firewall prevede la presenza di:

1. *Screening router*

È un router su cui siano state attivate funzionalità di filtraggio del traffico per impedire il passaggio dei protocolli o degli indirizzi di rete considerati non sicuri.

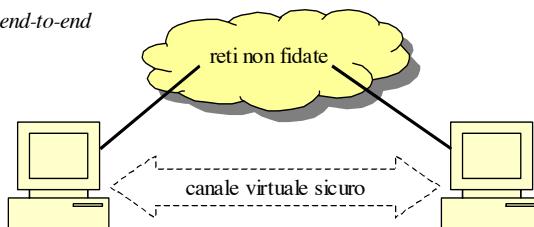
2. *Security gateway o security proxy*

Sono dei sistemi che permettono il passaggio di alcuni tipi di comunicazione di rete in base ad alcune caratteristiche dell'agente.

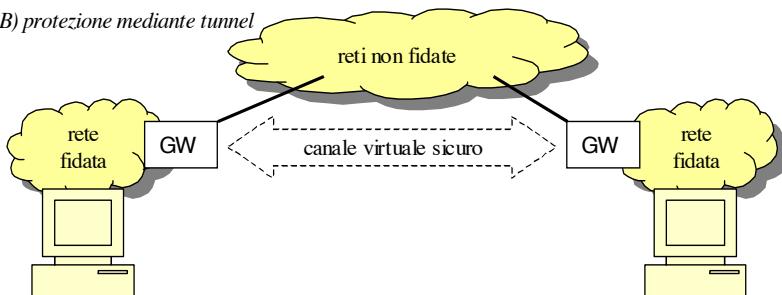
VPN

In generale i protocolli per le reti di calcolatori sono stati ideati senza prestare particolare attenzione alle loro caratteristiche di sicurezza. Normalmente è raro vedere attivate protezioni al livello 2: le protezioni di rete vengono applicate solitamente a livello 3, che è il primo livello in cui i pacchetti sono indirizzati direttamente dal mittente al destinatario. Si hanno due possibili soluzioni:

A) protezione end-to-end



B) protezione mediante tunnel



- ◆ **Sicurezza end-to-end**

In questo caso, si crea un canale logico sicuro direttamente tra i due nodi in comunicazione .

- ◆ **Tunnel**

Questa soluzione prevede che il canale sicuro venga stabilito tra due gateway (tipicamente due router). Questi tipi di protezione sono spesso chiamati VPN (Virtual Private Network). Le soluzioni molto spesso sono di tipo proprietario e scarsamente interoperabili. L'unico standard ufficiale in questo settore è costituito dall'architettura IPsec per la protezione delle reti TCP/IP applicabile sia a IPv4 che a reti IPv6.

- ◆ **Lo standard IPsec**

IPsec lascia intatto l'header dei pacchetti IP, per garantire compatibilità con l'attuale architettura di routing, e realizza le funzioni di sicurezza modificando il payload in due possibili formati diversi:

1. **Formato AH (Authentication Header)**

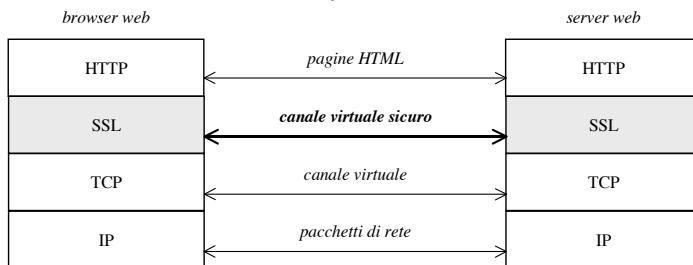
Questo formato fornisce funzioni di integrità, protezione da replay ed autenticazione del mittente aggiungendo al pacchetto un keyed-digest con chiave segreta ed un sequence number. La protezione include tutto il payload ed i campi fissi dell'header IP.

2. **Formato ESP (Encapsulating Security Payload)**

Il formato ESP fornisce le stesse funzioni di AH, più la riservatezza, perché il payload può essere cifrato con un algoritmo simmetrico, dopo avergli aggiunto un keyed-digest ed un sequence number. In questo caso però la protezione esclude completamente l'header che non può essere cifrato perché altrimenti i router non potrebbero leggere i campi necessari all'instradamento.

Sicurezza nel Web: uso di SSL

Il linguaggio HTML e il protocollo HTTP sono stati sviluppati, senza porre enfasi sugli aspetti di sicurezza. Quando si è provato ad usare il web per azioni commerciali, questa mancanza è subito emersa e per evitare di dover cambiare tali standard, Netscape ha scelto di creare uno strato intermedio di sicurezza, posizionato a livello sessione: SSL (Secure Socket Layer).

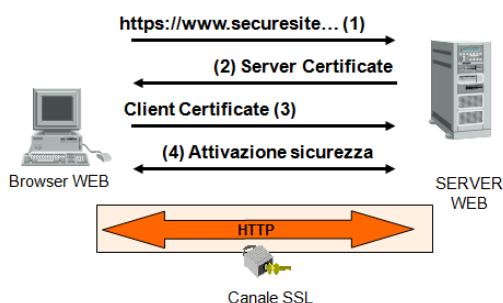


SSL è un protocollo per la crittografia di canale, che garantisce le proprietà di autenticazione, integrità e riservatezza. In particolare, SSL instaura un canale crittografico sicuro tra il layer di trasporto e quello applicativo. Tale protocollo supporta molti protocolli applicativi (http, nntp, smtp, ftp, telnet, ...).

SSL 2.0 prevedeva solo l'autenticazione Server, mentre SSL 3.0 prevede anche l'autenticazione Client.

Il protocollo SSL prevede le seguenti operazioni:

1. Il server deve essere dotato di un certificato X.509, perché la sua autenticazione viene ottenuta tramite una sfida asimmetrica propostagli dal browser, basata sulla chiave pubblica contenuta nel certificato.
2. Il server può optionalmente richiedere l'autenticazione del client; in questo caso, anche l'utente deve essere dotato a sua volta di un certificato X.509.
3. Una volta superata la fase di autenticazione, inizia la trasmissione dei dati che vengono suddivisi in blocchi. Ogni blocco è protetto mediante l'aggiunta di un identificativo numerico sequenziale di un keyed-digest e dalla cifratura simmetrica.
4. Gli algoritmi e le chiavi necessarie per queste operazioni sono negoziati all'atto dell'apertura della sessione SSL.



Ad esempio, nell'uso di http sicuro, SSL si attiva sulla porta TCP/443 e gli viene dedicata la URL <https://...>.

7. Politiche e meccanismi di sicurezza

Concetto di politica e di meccanismo di sicurezza

- ◆ *Politica*

Una politica è un insieme di linee guida per l'impostazione della sicurezza.

- ◆ *Meccanismo*

Un meccanismo è un insieme di dispositivi (hw e/o sw), che realizzano le politiche. Una politica può essere implementata da più meccanismi.

Politiche e meccanismi per la protezione del Web server

Le politiche che devono essere adottate per proteggere il web server sono quelle di seguito elencate. Per ciascuna sono poi indicati i principali meccanismi che si possono adottare per la loro applicazione.

- ◆ *Dare minimi privilegi ai processi attivati dal server (need-to-know)*

Il processo principale, che si pone in ascolto sulla porta 80, è purtroppo attivato dagli attuali S.O. con privilegi massimi (root in Unix, administrator in NT). Questo processo è il responsabile dell'attivazione dei processi figli inerenti i servizi offerti dal server Web. Questi servizi devono essere attivati associandoli ad un utente virtuale a cui siano assegnati minimi privilegi, così da impedire a eventuali processi con bug di aggredire l'intero sistema.

- ◆ *Proteggere il file system del server*

1. Agli utenti connessi al server web devono essere concessi privilegi minimi (read per le pagine html, execute per gli script o le applicazioni).

È buona norma impedire agli utenti la visualizzazione e la navigazione delle cartelle contenenti pagine html, script e applicativi. Non conoscere la versione e le informazioni degli script presenti sul server, impedisce lo sfruttamento dei loro possibili bug a scopo fraudolento

2. Ai Web administrator sono concessi anche i privilegi di modifica sulle pagine html e sui file di configurazione (write).

- ◆ *Confinamento dei singoli servizi*

È consigliata l'adozione di un server da dedicare esclusivamente a tale scopo, spostando ogni altro servizio o applicativo su altri elaboratori. Nel caso ciò non sia possibile e servizi diversi utilizzino il medesimo file system, è buona prassi adottare una politica di security volta a gestire separatamente gli account di ciascun servizio limitandosi a fornire ai singoli utenti i soli privilegi strettamente necessari.

Per i server Web è consigliabile inoltre tenere separati i path contenenti rispettivamente le pagine html e gli script (fornendo poi diritti di esecuzione limitati a questo path).

- ◆ *Attivare gli script con il minimo livello di privilegi*

Gli script devono essere attivati con gli stretti privilegi necessari a compiere la loro funzione.

- ◆ *Auditing del server ad un livello di granularità fine*

Si devono attivare tutte le forme di auditing possibili (meccanismi di valutazione) e tenere sotto costante monitoraggio i vari file di LOG. Si devono inoltre applicare al server tutti gli aggiornamenti e le patch che si rendano necessari.

Sicurezza della macchina su cui viene eseguito il browser

Occorre mettere in atto politiche e meccanismi anche riguardanti i calcolatori utilizzati per accedere mediante browser Web al WIS. In particolare, si dovrà avere:

- ♦ ***Protezione contro la violazione della privacy mediante cookies***

Nell'uso dei cookies, è possibile preservare la privacy disabilitandoli (ma alcuni siti non sarebbero più accessibili). Inoltre, esistono dei gestori di cookies che ne filtrano il contenuto rendendo disponibili ai server web solo alcune delle informazioni contenute.

- ♦ ***Protezione dall'esecuzione di componenti attivi fraudolenti: ActiveX***

Il client deve inoltre proteggersi dall'esecuzione di controlli ActiveX, Script, Applet di tipo malware. Possono inoltre esserci malware all'interno di plug-in (es.: documenti PDF artefatti), e si hanno altre minacce come i Trojan Horse.

Vediamo in particolare la protezione da ActiveX fraudolenti. Il termine *ActiveX* è usato per indicare un insieme di tecnologie e servizi per lo sviluppo di applicazioni basate su componenti riutilizzabili. È una tecnologia fondata sul modello COM (Component Object Model) di Microsoft e sulle sue estensioni (DCOM), che permette lo sviluppo di componenti client, server e di middleware in un'ottica di programmazione in ambiente distribuito.

I controlli ActiveX sono oggetti COM utilizzati come componenti run-time (che forniscono classi di oggetti e funzioni), oppure come interfaccia GUI (da utilizzare in applicativi e web).

Gli ActiveX sono usati per realizzare pagine WEB attive (e vengono scaricati tramite browser), e possono richiamare le API di sistema.

I controlli ActiveX non hanno un modello di sicurezza intrinseco, ma vengono presi "a scatola chiusa". Solo le *credenziali* che questi portano con sé (certificati e firma elettronica) vengono valutate per decidere se accettare o no il componente: l'utente può definire le proprie politiche di sicurezza (scegliere cioè quali componenti attivare e scaricare a diversi livelli di granularità). Si può quindi dire che è sempre l'utente l'ultimo arbitro della propria security.

Per fare questo, l'utente dovrà configurare le impostazioni di sicurezza del browser. Prendiamo come riferimento Internet Explorer, che consente di:

- a) *Impostare zone di sicurezza*

Lo spazio degli indirizzi viene suddiviso in 4 zone di sicurezza, a ciascuna delle quali si associa un livello di sicurezza cui corrisponde un profilo di security personalizzabile dall'utente:

- 1) Area Internet
- 2) Area Intranet
- 3) Area siti attendibili
- 4) Area siti con restrizioni

- b) *Impostare per ogni zona una policy di sicurezza*

Per il livello di sicurezza definito dall'utente è possibile definire una propria *policy di security*. Su ogni componente è possibile associare con appositi tool un livello di security predefinito (High, Medium, Low). Solo sulle applet Java è possibile associare un *profilo di security* personalizzato, includendo richieste d'accesso esplicite (I/O, accesso a funzioni di S.O, r/w HD, etc.).

Si possono impostare i 3 diversi livelli di sicurezza sul browser e sui singoli componenti.

Per decidere se accettare un ActiveX si può allora seguire la seguente procedura:

1. Si identifica la zona di sicurezza a cui appartiene il server da dove il componente è scaricato (in base all'indirizzo IP).
2. Si utilizza il livello di security definito per tale zona.
3. Nel caso dei 3 livelli predefiniti (High, Medium, Low) il comportamento è:

| | | File CAB firmato con questo livello | | |
|------------|--------|-------------------------------------|------------------------------|---------------------------|
| | | HIGH | MEDIUM | LOW |
| Zone Level | HIGH | Funziona in High | Query (Si = Funz. in Medium) | Query (Si = Funz. in Low) |
| | MEDIUM | Funziona in High | Funziona in Medium | Query (Si = Funz. in Low) |
| | LOW | Funziona in High | Funziona in Medium | Funziona in Low |

In Internet Explorer i certificati di autori ritenuti attendibili sono inseriti in un repository

I controlli ActiveX hanno alcuni svantaggi:

1. Non si è assicurati sui bugs e sul codice malizioso contenuto nei controlli scaricati.
2. I controlli possono chiamare API di sistema, e questo è un grande rischio.
3. Possono contenere cavalli di troia o possono instaurare back-door.
4. L'assegnazione dei siti alle zone di security in IE avviene in base all'indirizzo (address spoofing).
5. A causa dell'incapsulamento del codice, non si può testare la bontà del controllo.
6. I controlli ostili possono interferire con applicazioni server (COM/DCOM) quali gestori di transazioni o business critical.
7. Si è costretti a firmare il software per utilizzarlo in IE.

La firma del codice avviene aggiungendo al codice Java una serie di istruzioni per accedere alle Java Capabilities API del browser. Le Java Capabilities API permettono di definire:

- a. Chi è abilitato a svolgere una data azione (identificato dalla firma elettronica apposta in seguito) - *Class Principal*.
- b. Con che privilegi – *Class Privilege*
- c. su quali risorse del sistema – *Class target*.

Si wrappa poi il codice in un file *.JAR e poi lo si firma con degli appositi tool (bisogna possedere una chiave privata e la rispettiva chiave pubblica con il certificato della CA che l'ha validata). A questo punto, l'applet è pronta per essere scaricata.

Il modello di sicurezza Java

Le Applet Java adottano un modello di sicurezza denominato *sandbox*, al quale abbiamo già accennato in precedenza. Esso prevede 3 controlli che devono essere tutti superati perché l'applet possa essere eseguita dal browser:

- ♦ ***Controllo del bytecode Java (Bytecode verifier) contenuto nei file *.class***

Il bytecode verifier verifica il codice binario alla ricerca di comportamenti illeciti. Viene controllato il formato di ogni porzione di codice, controllando gli oggetti istanziati, l'incremento dei puntatori e le verifica restrizioni d'accesso.

Il bytecode ha una semantica più ricca di Java: la si può sfruttare per operazioni non ammesse dal linguaggio (minaccia).

- ♦ ***Controllo delle classi Java caricate in memoria (Class Loader)***

Il class loader determina se, in run-time, può essere aggiunta una classe all'ambiente Java. Le classi sono caricate in zone di memoria separate (namespace). Le classi fondamentali del core Java si trovano in namespace a cui sono assegnati i massimi privilegi (classi protette da corruzioni esterne).

Tutte le classi inserite nella variabile d'ambiente classpath acquisiscono massimi privilegi (non sono controllate).

- ♦ ***Controllo della correttezza dei metodi secondo un set di regole (Security Manager)***

Il security manager controlla i metodi prima che siano eseguiti, basandosi sulla classe di provenienza e su un set di regole predefinite.

In particolare, il security manager controlla:

1. Conflitti nell'assegnazione dello spazio dei nomi.
2. Le chiamate ai processi del SO.
3. L'accesso alle classi.
4. Operazioni di lettura/scrittura su HD.
5. Operazioni di I/O su socket.

Se una condizione viene violata, allora si solleva una Security Exception. Ogni browser implementa una propria versione del security manager.

8. Controllo dei dati

La sicurezza in un DBMS

Diamo ora per scontata la presenza di barriere che si interpongono nell'accesso ai dati, i quali si troveranno sempre al livello di massima protezione all'interno della rete, e concentriamoci sull'accesso ai dati. Naturalmente, i dati sono gestiti da un DBMS, il quale deve soddisfare i seguenti requisiti:

1. Deve fornire un meccanismo di autorizzazione dinamica (mediante grant/revoke).
2. Non deve presentare delle back doors.
3. Deve avere delle prestazioni ragionevoli.
4. Deve consentire diversi livelli di granularità nell'accesso ai dati.
5. Deve effettuare diverse tipologie di controllo (accesso, inferenza, flusso).

I controlli di sicurezza del DBMS

Come appena accennato, il DBMS deve effettuare vari tipi di controlli:

- ♦ ***Controlli di flusso***

Sono i controlli che consentono di evitare che le informazioni fluiscano in modo da raggiungere utenti non autorizzati, a partire da utenti autorizzati.

Questo tipo di controllo richiede che venga effettuata un'analisi del codice.

- ♦ ***Controlli di inferenza***

Sono controlli che evitano che le informazioni riservate vengano dedotte da utenti non autorizzati, mediante l'utilizzo di una serie di interrogazioni lecite:

1. *Accesso indiretto*

Ad esempio, se l'utente non è autorizzato a visualizzare Y, può eseguire una query del tipo:

SELECT X FROM r WHERE Y = value

Per dedurlo in maniera indiretta (supponendo che sia autorizzato a visualizzare X).

2. *Dati correlati*

Se ad esempio si hanno dei dati legati dalla relazione $z = t * k$ dove t e k sono visibili mentre z non è visibile, naturalmente z può facilmente essere calcolato a partire da t e k.

3. *Dati mancanti*

È possibile che ad un utente sia consentito solo vedere i dati statistici relativi ad un insieme di dati.

Ad esempio, si può sapere quanti dipendenti hanno uno stipendio superiore ad un certo valore. È però naturale che se si trova un valore per il quale si ha un solo dipendente con stipendio superiore, si individuerà immediatamente chi è quel dipendente (quello alla posizione di maggiore responsabilità), e con una serie di interrogazioni si può determinare con esattezza il suo stipendio.

Il problema dell'inferenza statistica è prevenuto mediante la perturbazione dei dati e il controllo sulle query (si impone una dimensione minima dei risultati restituiti dalla query).

- ♦ ***Controlli di accesso***

I controlli di accesso sono quelli su cui ci concentreremo maggiormente. Per effettuare tali concetti, si definiscono le varie autorizzazioni da attribuire ai singoli soggetti sui vari tipi di oggetti.

Le autorizzazioni potranno essere di vario tipo (lettura, scrittura). Gli utenti sono suddivisi in vari livelli e in varie gruppi o classi. Ogni utente può appartenere a più di un gruppo.

Gli utenti vengono anche raggruppati in amministratori, utenti finali e programmati applicativi.

A riguardo degli oggetti, è fondamentale scegliere opportunamente la granularità secondo la quale vengono definiti (tabella, colonna, tupla, ...), perché da essa dipendono sia l'efficacia dei controlli, sia la loro efficienza. Si noti che occorre proteggere anche gli schemi e i dizionari.

Definizione di regole di accesso al DBMS (regole a 3 componenti)

Le regole di accesso di base di un DBMS sono specificate come triple del tipo:

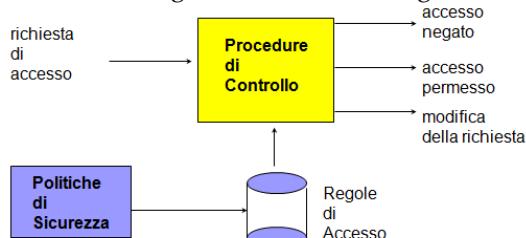
< soggetto, oggetto, permessi >

Questo modello può poi essere esteso con l'aggiunta di un quarto elemento:

< soggetto, oggetto, permessi, vincolo >

Questo quarto elemento rappresenta un vincolo riguardante il contenuto. Tale vincolo viene espresso mediante una *protection view*.

Il modello adottato per l'applicazione delle regole di accesso è il seguente:



Quindi le procedure di controllo ricevono in ingresso le regole di accesso e le richieste di accesso: la procedura stabilisce poi, sulla base delle politiche, se l'accesso richiesto viene consentito oppure negato; in alternativa, le procedure di controllo possono modificare la richiesta ricevuta. Si parla in questo caso di *query modification*. Ad esempio, la query:

`SELECT * FROM EMP`

Può essere trasformata dal DBMS in:

`SELECT * FROM EMP WHERE Stipendio < 1500`

Diverse politiche di sicurezza: sistema aperto e sistema chiuso

Si possono adottare due diverse tipologie di politiche di sicurezza

- ♦ **Sistema aperto**

In un sistema aperto, quando si riceve una richiesta si ricercano delle regole di accesso che impediscono l'accesso richiesto. Se esistono regole di questo tipo, l'accesso viene negato; in caso contrario, il DBMS permette che l'accesso avvenga.

- ♦ **Sistema chiuso**

In un sistema chiuso, quando si riceve una richiesta si ricercano delle regole di accesso che consentono l'accesso richiesto. Se esistono regole di questo tipo, l'accesso viene consentito; in caso contrario, il DBMS nega l'accesso.

La differenza è quindi che in un sistema aperto le regole dicono quando l'accesso deve essere negato, in un sistema chiuso invece le regole vengono usate per specificare quando l'accesso è consentito.

Controllo di tipo discrezionale (DAC)

Nel controllo discrezionale, gli utenti amministrano i dati che possiedono (concetto di proprietario). Il proprietario dei dati può anche autorizzare altri utenti all'accesso, definendo il tipo di accesso da concedere loro (lettura, scrittura, esecuzione) ed eventualmente consentendo solo accessi selettivi (basati su nome, contenuto, parametri di sistema, storia, aggregazione di dati).

In un DBMS, se si vogliono fornire o revocare le autorizzazioni secondo la politica DAC, si devono utilizzare le istruzioni SQL3 denominate GRANT e REVOKE. La gestione dei permessi è associata ai ruoli, ed è possibile fornire un certo ruolo ai singoli utenti. In particolare, potremo scrivere:

```

GRANT lista permessi ON Tabella TO NomeroUolo
GRANT NomeroUolo TO Utente
GRANT NomeroUolo TO NomeroUolo
REVOKE NomeroUolo FROM Utente
  
```

Controllo mandatorio (MAC)

Per le basi di dati contenenti dati sensibili, il controllo discrezionale non è sufficiente, perciò si adottano politiche basate sulla classificazione dei dati e degli utenti, dette MAC (Mandatory Access Control) o politiche a controllo di tipo mandatorio.

Queste politiche prevedono che tutti i dati e tutti i soggetti vengano classificati in opportuni *livelli* (nel caso di dati, detti *livelli di sensitività* e nel caso dei soggetti detti *livelli di clearance*).

Si hanno inoltre delle categorie (ad esempio: Produzione, Personale, ...). Si definiscono poi delle *classi di sicurezza*, che sono delle coppie del tipo:

< livello di classificazione, insieme di categorie >

In questo modo, si attribuisce alle categorie dell'insieme quel particolare livello di classificazione.

Le classi di sicurezza devono essere legate tra loro da una relazione di ordinamento (anche parziale). Ad esempio, una possibile classificazione è quella che prevede 4 livelli di classificazione (sensibilità e clearance), che sono Unclassified, Confidential, Secret, Top Secret.

I meccanismi di sicurezza devono garantire che tutti i soggetti abbiano accesso solo ai dati per cui possiedono la clearance appropriata (regole *soggetti-oggetti*). Nel dettaglio, le regole che si applicano sono 2 e, combinate tra loro, impediscono che si abbiano flussi di dati tra soggetti *high* e *low*:

- ♦ ***Regola no read up***

Un soggetto S è autorizzato ad accedere in lettura a un oggetto O solo se $L(S) \geq L(O)$.

- ♦ ***Regola no write down***

Un soggetto S è autorizzato ad accedere in scrittura a un oggetto O solo se $L(S) \leq L(O)$.

Tipi di controllo di accesso

- ♦ ***Accesso basato sul nome***

L'accesso ai dati basato sul nome prevede che diversi utenti possano accedere a campi diversi di una tabella. Si definiscono quindi tutti i nomi dei campi ai quali un utente può accedere.

- ♦ ***Accesso basato sul contenuto***

L'accesso ai dati basato sul contenuto prevede invece che un utente possa accedere solo ad alcuni campi e solo ad alcune tuple, sulla base dei valori contenuti nelle tuple. Ad esempio, in una tabella con l'elenco dei dipendenti che abbia il campo "Reparto", si può impostare ad un utente di visualizzare solo le tuple relative ai dipendenti di un certo reparto.

Per fare ciò, si usano le viste e si attribuisce all'utente il permesso di accedere alla vista, ma non alla tabella originale.

- ♦ ***Controllo basato sul tipo di accesso***

Il controllo basato sul tipo di accesso prevede che alcuni utenti possano effettuare solo alcune operazioni sui dati (ad esempio, solo la lettura e non la scrittura).

Definizione di regole di accesso al DBMS (regole a 6 componenti)

Le regole di accesso al DBMS possono essere definite non solo mediante delle triple, come già discusso, ma anche mediante delle tuple a 6 componenti, del tipo:

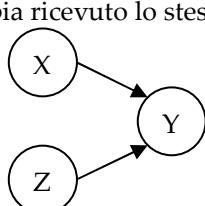
< autorizzatore, autorizzato, risorsa protetta, privilegio di acceso, propagazione successiva >

Dove:

1. Autorizzatore è l'amministratore del sistema o un proprietario della risorsa protetta;
2. Autorizzato è colui che riceve, in modo discrezionale (tramite GRANT) il privilegio sulla risorsa.
3. La risorsa protetta è l'oggetto relativamente al quale si sta fornendo il permesso di accesso.
4. Privilegio di accesso è la modalità secondo la quale è consentito l'accesso (lettura, scrittura, ...).
5. Propagazione successiva è la possibilità per l'autorizzato di propagare mediante GRANT il privilegio ottenuto. In particolare, i valori ammessi sono:
 - a. Nessuna propagazione: solo privilegio USE.
 - b. Propagazione incondizionata (UP): privilegi USE e PROPAGATE.
 - c. Propagazione limitata (BP): orizzontale/verticale.

Il problema della revoca

Come abbiamo detto, è possibile anche che i permessi vengano revocati. In tal modo, si ha un problema legato alla possibilità di propagare i diritti: se X ha fornito un permesso a Y e si sceglie di revocare il diritto ad X, allora tale diritto dovrebbe essere revocato anche ad Y. Si parla allora di revoca ricorsiva (*cascading*). Tuttavia, è possibile anche che l'utente Y abbia ricevuto lo stesso permesso anche da un altro utente Z:



In questo caso, se X revoca l'autorizzazione a Y, come ci si deve comportare? Per risolvere questo problema, si usano dei timestamp: ad ogni autorizzazione (cioè ad ogni "freccia") si associa un timestamp, che indica l'istante in cui il permesso è stato concesso. Se Z ha concesso il permesso ad Y in un istante successivo rispetto all'istante in cui X ha dato il permesso a Y, allora Y mantiene il proprio permesso, altrimenti no.

Sss-a<DS

Vulnerabilità nell'accesso ai dati

- ♦ *Furto di password*

Quando si usa un database, solitamente l'accesso viene fornito mediante l'uso di un nome utente e di una password. Nella maggior parte dei casi, tali password vengono codificate all'interno del codice, ma questa scelta è molto pericolosa, perché nel caso in cui qualcuno dovesse riuscire ad accedere agli script, si garantirebbe accesso completo alla base di dati, anche mediante l'uso del Web.

La soluzione migliore è quindi quella di memorizzare il nome utente e la password in un file al di fuori dell'albero di directory accessibile mediante il Web.

- ♦ *SQL Injection*

Un'altra tecnica di attacco alla base di dati è quella detta *SQL Injection*, che consiste nell'alterare gli statement SQL mediante la manipolazione degli input forniti.

Le applicazioni sono soggette a questo tipo di attacco quando i programmati accettano gli input dell'utente e li inseriscono direttamente all'interno degli statement SQL, senza filtrare eventuali caratteri pericolosi.

In tal modo, si potranno verificare non solo furti di dati dal database, ma anche alterazioni e cancellazioni dei dati stessi: in tal modo, è possibile che risulti compromessa l'intera macchina.

La contromisura principale è quella di validare gli statement SQL dinamici.