

Modelli e processi aziendali

Breve presentazione dei principali modelli utilizzati nelle aziende



Indice

<u>1 Metodi diversi per la gestione dei processi.....</u>	<u>3</u>
<u>1.1 Il Total Qualità Management (TQM).....</u>	<u>3</u>
<u>1.2 Business Process Improvement - BPI.....</u>	<u>4</u>
<u>1.3 Business Process Reengineering - BPR.....</u>	<u>5</u>
1.3.1 Michael Hammer e James Champy.....	6
1.3.2 Un metodo in cinque punti: Business Process Reengineering.....	6
1.3.3 Circostanze generiche che aiutano a capire quando il BPR è consigliabile.....	7
1.3.4 Critiche al metodo BPR.....	7
1.3.5 BPR confrontato al Kaizen.....	8
<u>1.4 SIX SIGMA.....</u>	<u>9</u>
<u>2 Il Total Cost of Ownership - TCO.....</u>	<u>11</u>
<u>2.1 Elementi di criticità del TCO.....</u>	<u>12</u>
<u>2.2 Efficienza ed efficacia.....</u>	<u>12</u>
<u>2.3 Ma il valore è un'altra cosa.....</u>	<u>12</u>
<u>2.4 Tre consigli per fare ordine.....</u>	<u>13</u>
<u>3 UML: Introduzione.....</u>	<u>15</u>
<u>3.1 L'Analisi ed i Programmatori: un'incompatibilità atavica</u>	<u>15</u>
<u>3.2 Non è vero.</u>	<u>15</u>
<u>3.3 Un po' di storia</u>	<u>16</u>
<u>3.4 Primo contatto con l'UML</u>	<u>17</u>
<u>3.5 Gli strumenti descrittivi</u>	<u>18</u>
<u>3.6 Conclusioni</u>	<u>20</u>
<u>3.7 UML: Use Case</u>	<u>20</u>
<u>3.8 I requisiti utente</u>	<u>21</u>
<u>3.9 Use Case ragionato</u>	<u>24</u>
<u>3.10 Conclusioni</u>	<u>25</u>
<u>Bibliografia.....</u>	<u>26</u>

1 Metodi diversi per la gestione dei processi

Negli ultimi anni (approssimativamente dal 1980), sono stati sviluppati da consulenti, manager e guru dell'economia, diverse metodologie per la gestione dei processi. E' possibile affermare che si tratta essenzialmente di punti di vista differenti. Infatti la maggior parte, se non tutti questi metodi, presuppongono un orientamento al processo da parte delle organizzazioni, parola d'ordine sempre più imperante nei modelli esistenti oggi.

Tuttavia la banale applicazione di un modello da sola non serve a risolvere complessi problemi aziendali, soprattutto se non è il management in primis a credere in un cambio di filosofia e gli "opinion maker" aziendali non vengono messi nelle condizioni di piena operatività tramite il pieno appoggio dell'alta dirigenza. Ogni cambiamento in azienda richiede un forte e motivato appoggio aziendale, che deve essere il primo passo verso il cambiamento, non solo di facciata, ma profondo, e deve portare a interiorizzare i modelli culturali prima ancora che le tecniche di applicazione.

Le sole tecniche, infatti, ci aiutano nel lavoro ordinario, sono di stimolo per svolgere al meglio il nostro compito, ma da sole non bastano a operare quel salto che, invece, troppo spesso l'azienda si augura.

Per tornare a questa breve panoramica, ogni modello ha un preciso campo di applicazione, proprie modalità di applicazione e deve sempre essere ampiamente supportato dalle funzioni aziendali direttive, se non se ne vuole minare l'efficacia da subito. Adesso una breve panoramica e i relativi campi di applicazione.

1.1 Il Total Quality Management (TQM)

E', forse, il metodo più vecchio per la gestione del processo orientato al cliente. Il concetto implementato da Kaoru Ishikawa, già nel 1950, parte dalla considerazione che "il processo dopo il tuo è il tuo cliente".

Prevenire, quanto prima possibile difetti ed errori in un processo, è alla base del Total Quality Management. Ciò permette di fornire prodotti e servizi che soddisfino il cliente, al costo più basso possibile, infatti spostando il controllo all'inizio del processo produttivo si abbattano in modo significativo gli scarti di produzione. Il compito più importante per il TQM è dunque concentrarsi nell'assicurazione della qualità dei processi e dei prodotti/servizi, attuandola in maniera tale da migliorare la qualità e la soddisfazione del cliente interno (si può considerare cliente anche un collega, nel momento in cui partecipiamo allo stesso processo aziendale) ed esterno, in maniera persistente e continua, d'accordo con le richieste e le aspettative degli stakeholders (clienti, fornitori, collaboratori ed azionisti).

La tendenza, e l'auspicio aziendale, è di spostare talmente in avanti il controllo del processo, da intervenire fin dalle fasi di progettazione di un nuovo prodotto.

L'implementazione corretta del total quality management produce una migliore soddisfazione del cliente, elimina difetti e scarti, migliora l'impegno e la motivazione nel personale, fa crescere

produttività e competitività (sembra incredibile, ma sapere di poter fare meglio una cosa ha un benefico effetto anche sui lavoratori).

Diversi sono gli specialisti della qualità che hanno sviluppato concetti e metodi sul total quality management. I più noti, a parte K. Ishikaw, sono W. Deming, J. Juran and P. Crosby. Ciascuno di essi ha enfatizzato diversi aspetti del TQM, ma il loro messaggio è comune a tutti: **"l'intera organizzazione, guidata da una leadership completamente coinvolta, deve essere impegnata al miglioramento della qualità, che è un processo continuo e che non ha termine. L'obiettivo deve essere la soddisfazione delle richieste del cliente attraverso il miglioramento dei processi. I processi saranno migliorati prevenendo i problemi e risolvendoli sistematicamente e continuamente"**.

Crosby in particolare concentra l'attenzione sui costi della qualità. L'enfasi è posta sulla pianificazione per il raggiungimento di zero difetti e sul fare la cosa giusta la prima volta (Crosby 1979).

Juran paragona la gestione della qualità alla gestione economico/finanziaria dell'azienda. In ambedue i processi di gestione esistono tre stadi; nel processo economico/finanziario i tre stadi sono il budget, il controllo e la conseguente riduzione dei costi. Nel processo di gestione della qualità gli stadi corrispondenti sono la pianificazione della qualità, il controllo della qualità e il miglioramento continuo della qualità (Juran 1989).

Deming ha creato l'ormai ben noto circolo PDCA (Plan-Do-Check-Act). Il concept è pianificare il miglioramento della qualità, portare avanti il piano, verificare i risultati e ricominciare da capo. In sintesi **il miglioramento della qualità è un processo continuo**.

Sono i processi che costituiscono i prodotti o i servizi che devono soddisfare i clienti: lavorare dunque con i processi è la chiave del miglioramento continuo. La natura ripetitiva dei processi fa in modo che il miglioramento sia continuo.

1.2 Business Process Improvement - BPI

Si tratta di un metodo molto efficace per il miglioramento dei processi che produce, per una qualsiasi organizzazione, una maggiore efficienza. L'applicazione del metodo semplifica le attività del processo eliminando sprechi ed inutili passaggi burocratici.

Mentre il Total Quality Management ha un approccio al processo dal basso dell'organizzazione (diremmo Bottom Up), l'applicazione della metodologia BPI inizia contemporaneamente dall'alto (Top-Down) e dal basso (Bottom – Up) dell'organizzazione. Nel BPI i manager e gli specialisti hanno un ruolo importantissimo per il miglioramento del processo. La base di partenza per l'applicazione del BPI è data dai seguenti elementi significativi:

- Il coinvolgimento totale del top management
- L'impegno per una attività di lungo e medio termine
- L'utilizzazione di metodi ben definiti
- L'identificazione dei process owners (chi gestisce i processi)
- Lo sviluppo di misurazioni di risultati e di un sistema di reporting
- La definizione dei processi

Attualmente, in special modo nelle organizzazioni medie e grandi, esistono diversi gruppi di lavoro dedicati al miglioramento dei propri sottoprocessi. Generalmente tali gruppi lavorano con entusiasmo e misurano le proprie performance. Il problema cruciale è che questi non conoscono l'effetto che il loro lavoro ha sulle altre attività del processo. La conseguenza è che spesso i sottoprocessi risultano ottimizzati **ma il processo nella sua interezza non è stato migliorato**. L'obiettivo che il BPI si prefigge è quello di rendere il processo nella sua interezza efficace ed adeguato (visione **olistica** o **sistemica**). Ciò si traduce nel raggiungimento dei risultati richiesti e nel ridurre le risorse necessarie. Occorre dunque fare in modo che i processi siano flessibili in modo da essere adattabili alle continue e variabili esigenze del mercato e della stessa organizzazione. Altri obiettivi sono l'eliminazione degli errori, la riduzione dei ritardi, la crescita della conoscenza, la facilità d'uso, la fedeltà dei clienti, tutto ciò allo scopo di fornire vantaggi competitivi all'organizzazione.

Nelle organizzazioni complesse il BPI si sviluppa, in generale, in cinque fasi:

1. **Organizzazione per il miglioramento.** Durante questa prima fase il top management apprende la metodologia BPI, seleziona i processi critici, e nomina i process owner per ciascun processo selezionato. I process owners organizzano i gruppi di lavoro per il miglioramento del processo (Process Improvement Team - PIT) che stabilisce i vincoli, sceglie i metodi di misura, identifica gli obiettivi di miglioramento e sviluppa il piano di progetto (project plan).
2. **Comprensione del processo.** Durante questa fase il gruppo di lavoro (PIT) produce la mappa del processo attuale, analizza il rispetto delle procedure esistenti, raccoglie tutte le informazioni disponibili (costo, tempi, ecc.) ed allinea le attività correnti alle procedure. Scopo di tale fase è quello di raggiungere la conoscenza dettagliata del processo.
3. **"Fluidificazione" del processo.** Si tratta in questa fase di rendere il processo realmente fluido, vale a dire senza ostacoli durante il suo corso di azione. Non si tratta solo di semplificare ma di rimuovere tutte le attività che non aggiungano valore. E' in questa fase del BPI che la creatività e la competenza del gruppo di progetto viene effettivamente messa alla prova.
4. **Implementazione, misure e controllo.** Il processo (migliorato) viene in questa fase realizzato (messo in funzione), i sistemi di misura ed i controlli vengono stabiliti. E' indispensabile avere in funzione un efficiente sistema di report, in modo da poter attuare in tempi brevi tutte le modifiche necessarie.
5. **Miglioramento continuo.** Non bisogna mai perdere di vista il fatto che qualsiasi processo è migliorabile. E' questo ora il compito degli effettivi operatori del processo.

1.3 Business Process Reengineering - BPR

Il metodo di **Business Process Reengineering**, o **BPR**, è descritto da Hammer e da Champy come "il riesame fondamentale e la riprogettazione radicale dei processi organizzativi, per realizzare il **miglioramento drastico** delle prestazioni correnti di una Azienda / settore, riguardo il costo, i servizi, etc.

Piuttosto che organizzare una ditta specializzandola nelle sue attività funzionali (come la produzione, la contabilità, la penetrazione sul mercato, ecc.) e guardare le mansioni che ogni funzione effettua, Hammer e Champy suggeriscono di guardare i processi completi. Dalla acquisizione dei materiali, alla produzione, all'introduzione sul mercato e alla capacità di

distribuzione. Dalla attenta analisi dei processi si evidenzieranno tutti quei processi che possono essere migliorati e, in taluni casi, addirittura eliminati.

Quindi bisogna innanzitutto scomporre le attività della ditta in una serie di processi che mettano in evidenza i processi critici della attività aziendale. Dalla conoscenza dei processi aziendali si potranno operare le semplificazioni e i miglioramenti che potranno portare non ad un miglioramento relativo di prestazioni, ma ad un drastico miglioramento, quale ad esempio, il dimezzamento dei tempi di progettazione (non il risparmio del 3% del costo di progettazione).

La creazione di valore per il cliente è il fattore principale per il BPR e la **tecnologia dell'informazione** svolge spesso un ruolo importante.

Anche per il BPR il coinvolgimento di importanti funzioni aziendali è il prerequisito per la completa riuscita del BPR.

1.3.1 Michael Hammer e James Champy

I fautori principali del re-engineering sono Michael Hammer e James Champy, che hanno illustrato il proprio metodo in una serie di libri, tra cui sono stati di particolare successo “Reengineering the Corporation”, “Reengineering Management e l'ordine del giorno”. I due sostengono che troppo tempo in azienda viene sprecato, utilizzato male; spesso le lentezze sono dovute alle mansioni che prevedono il passaggio di dati / informazioni da un reparto ad un altro. Hammer e Champy sostengono che è molto più efficiente nominare una squadra che effettua tutte le mansioni del processo, piuttosto che suddividere il processo tra più squadre e reparti

1.3.2 Un metodo in cinque punti: Business Process Reengineering

Davenport (1992) prescrive un metodo di cinque-punti per l'applicazione del modello di Business Process Reengineering:

1. **Sviluppare la visione del Business e gli obiettivi relativi:** La metodologia BPR è orientata da una visione di business che implica che gli obiettivi specifici del business, quali la riduzione dei costi, la riduzione dei tempi, il miglioramento della qualità dei prodotti, siano esplicitati fin dall'inizio.
2. **Identificare i processi di Business da ri-progettare:** la maggior parte delle ditte usano il metodo “high-impact,, che mette a fuoco i processi più importanti o quelli che sono in conflitto con la visione di Business. Poche ditte, invece, usano un metodo esaustivo per identificare tutti i processi all'interno di un'organizzazione e quindi assegnare le priorità nell'ordine della riprogettazione dei processi.
3. **Capire e misurare i processi attuali:** per evitare di ripetere i vecchi errori e fornire una linea di condotta per i miglioramenti futuri.

4. **Identificare i fattori di successo:** la consapevolezza della possibilità di riuscita del BPR, può e deve influenzare la riuscita del BPR.
5. **Progettare e sviluppare un prototipo del nuovo processo:** il disegno reale del processo non dovrebbe essere pensato come la fine del processo BPR, ma dovrebbe essere visto come prototipo, necessario per le ripetizioni successive. L'utilizzo di un prototipo allinea il metodo di Business Process Reengineering con le altre metodologie che prevedono la consegna rapida dei risultati e la partecipazione (e la soddisfazione) del cliente finale.

Esiste un sesto punto supplementare del metodo di BPR: consiste nell'adattare la struttura organizzativa ed il modello di controllo con il processo primario appena progettato.

1.3.3 Circostanze generiche che aiutano a capire quando il BPR è consigliabile

Anche se è difficile esprimere un parere sulla capacità del BPR di rispondere alle esigenze aziendali, alcuni fattori da considerare sono:

- La concorrenza surclassa chiaramente l'azienda?
- Ci sono molti conflitti all'interno dell'organizzazione aziendale?
- Le riunioni sono estremamente frequenti?
- C'è un uso eccessivo della comunicazione non-strutturata? (appunti, email, ecc)
- È possibile considerare un metodo di miglioramenti graduali e incrementali? (si veda: Kaizen).

1.3.4 Critiche al metodo BPR

Il metodo BPR si è guadagnato una pessima reputazione perché spesso i progetti BPR hanno innescato molti licenziamenti. Nonostante **la hype** che ha circondato l'introduzione del Business Process Reengineering, parzialmente dovuta al fatto che gli autori di "Reengineering the Corporation", secondo quanto riferito, hanno comprato moltissime copie del loro libro per raggiungere la parte alta della classifica dei bestseller più venduti, il metodo non ha interamente risposto alle relative aspettative aziendali. I motivi principali sembrano essere questi:

- Il BPR suppone che il fattore che limita le prestazioni dell'organizzazione sia dato dall'inefficienza dei relativi processi, ma allo stesso tempo il BPR non offre mezzi adeguati per convalidare questo presupposto.
- Il BPR presuppone la necessità di iniziare il processo di miglioramento delle prestazioni facendo tabula rasa di tutto il passato dell'azienda. Quindi ignora completamente lo status quo aziendale, e tutte le particolarità che invece andrebbero preservate.
- Il BPR non fornisce un modello efficace per concentrare gli sforzi di miglioramento sui vincoli dell'organizzazione (come trattato da Goldratt nel libro "Theory of Constraints")
- A volte, o forse abbastanza spesso, un cambiamento graduale ed incrementale (come nel Kaizen) può essere un metodo migliore.
- Il BPR è culturalmente orientato ai modelli aziendali degli Stati Uniti.

1.3.5 BPR confrontato al Kaizen

Paragonare il Kaizen al metodo di BPR equivale a confrontare due filosofie opposte. Il Kaizen è maggiormente orientato all'uomo, è generalmente più facile da mettere in pratica, ma richiede una disciplina e una applicazione di lungo periodo e fornisce solo un piccolo incremento verso il cambiamento. Il metodo di Business Process Reengineering, invece, è più duro da osservare, è maggiormente orientato alla tecnologia, permette un cambiamento radicale, ma richiede abilità considerevoli della gestione del cambiamento.

Alla luce di quanto detto finora, la re-ingegnerizzazione dei processi può essere definita nel seguente modo: "Ripensamento e **riprogettazione radicale** del **processo** per raggiungere un miglioramento **sostanziale** nella performance." In questa definizione ci sono quattro parole chiave su cui vale la pena di soffermarsi.

Cominciamo dal concetto di miglioramento "**sostanziale**". Lo scopo della reingegnerizzazione non è quello ottenere miglioramenti marginali per l'organizzazione, ma piuttosto quello di ottenere prestazioni aziendali eccellenti nella performance e di effettuare una svolta sostanziale nella performance.

La seconda parola chiave è "**radicale**". Radicale vuol dire andare alla radice delle cose. Non si tratta quindi di migliorare quello che già esiste. Si tratta di buttare via quello che già esiste e ricominciare da capo: reinventare il modo di compiere il lavoro.

La terza parola chiave è "**processo**". Per processo intendiamo un gruppo di attività interconnesse che insieme creano valore per il cliente. Nessuna di queste attività, prese singolarmente, hanno valore trascurabile per il cliente. Il cliente è interessato solo al risultato finale. Il processo è il cuore di qualsiasi tipo di organizzazione. E' per mezzo del processo che l'organizzazione crea valore per il loro cliente.

La quarta parola chiave è "**riprogettazione**". Si tratta di progettare come deve essere svolto il lavoro. La progettazione del processo è di primaria importanza. Un processo non ben progettato, anche se svolto da personale esperto e motivato, non potrà essere eseguito in maniera soddisfacente.

L'uso efficace della Information Technology nella riprogettazione dei processi è molto importante. La riprogettazione non può essere fatta a "piccoli passi": deve essere "**tutto o niente**" e condotta in modo tale da prevederne il più possibile i risultati. Il BPR va quindi utilizzato quando non c'è altra scelta; quando l'intera esistenza dell'organizzazione è minacciata.

La metodologia BPR richiede che i processi base siano considerati orizzontalmente. **I confini organizzativi devono essere abbattuti**. Uno dei sistemi più utilizzati è quello di costituire un gruppo di lavoro che comprenda appartenenti, possibilmente i responsabili, di ciascuna funzione attraversata dal processo. La prima operazione da compiere è l'analisi approfondita del processo esistente in modo da comprenderne a fondo lo scopo reale. Il gruppo di lavoro deve individuare quali fasi o attività del processo aggiungono effettivamente valore (Value Methodology) e

ricercare nuove strade per raggiungere l'obiettivo. Le domande da porsi sono: perché le cose vengono fatte in questo modo e che cosa succederebbe se fossero fatte in modo diverso.

L'iter suggerito per l'implementazione del BPR dovrebbe essere quindi:

1. Selezionare il processo
2. Identificare i fattori di cambiamento
3. Creare la "vision" del processo
4. Comprendere a fondo il processo
5. Progettare il nuovo processo

1.4 SIX SIGMA

Six Sigma è una "filosofia operativa" che può portare benefici ai clienti, agli azionisti, ai collaboratori e ai fornitori. In sintesi si tratta di una metodologia centrata sul cliente che elimina gli sprechi, eleva il livello della qualità e migliora, quindi, le performance finanziarie dell'organizzazione. In modo più specifico SS esamina a fondo i processi ripetitivi di progettazione, produzione, acquisti, servizi e organizzazione in modo da prevenire i difetti e gli errori.

L'obiettivo di Six Sigma è l'ottenimento di non più di 3,4 difetti o errori per milione di eventi nei processi di progettazione e produzione sia che si tratti di prodotti sia che si tratti di servizi.

Sigma è il simbolo utilizzato per identificare, in statistica, la misura della variazione in un processo. Più è alto il valore di Sigma minori sono le deviazioni dallo standard in un processo.

SS traduce le richieste del cliente in termini di operatività dell'organizzazione e definisce i processi critici e le attività che devono essere compiute in maniera eccellente, senza difetti. L'approccio SS deve essere intrapreso come un progetto per l'organizzazione. Si tratta di una svolta culturale.

Il progetto Six SIGMA può essere diviso in sei fasi:

1) Definizione del coinvolgimento del management.

- Questo vuol dire formare il management sugli strumenti ed i principi del sistema SS.

2) Raccolta delle informazioni.

- Le informazioni vanno raccolte con contatti intensi con clienti, fornitori e collaboratori.

3) Formazione.

- Tutti gli elementi dell'organizzazione devono essere formati sui principi e gli strumenti.

4) Sviluppo del sistema di monitoraggio.

- Le misure e gli indicatori vanno definiti.

5) Scelta del processo da migliorare.

- E' questa la fase di partenza per la realizzazione del progetto. E' necessaria una mappatura dei processi critici, una identificazione dei problemi, la eliminazione delle attività che non producono valore (Value Methodology) e, quindi, la definizione delle priorità.

6) Implementazione del progetto SS.

- Si tratta di definire il processo da migliorare e gli obiettivi che si vogliono raggiungere, stabilire le misure, analizzare e definire gli scostamenti tra "come è" e "come dovrebbe essere", migliorare il processo e validarlo per mezzo di simulazioni e statistiche, comunicare a tutta l'organizzazione i cambiamenti, istituzionalizzare, documentare e monitorare.

2 Il Total Cost of Ownership - TCO

TCO è un acronimo in lingua inglese che sta per "Total Cost of Ownership" (Costo totale di possesso). Il TCO è un approccio sviluppato da Gartner Group nel 1987, ed è stato originariamente utilizzato per calcolare tutti i costi del ciclo di vita di una qualunque apparecchiatura informatica, per problemi inerenti l'acquisto, l'installazione, la gestione, la manutenzione e lo smaltimento a fine ciclo vitale.

L'analisi TCO deve perciò tener conto di:

- Tutti i costi per l'acquisto dei componenti (Hardware e Software) compresa la ricerca del fornitore sul mercato, i costi di amministrazione per le ricerche di mercato, i costi delle licenze software, etc. ;
- costi per lo sviluppo di personalizzazione degli applicativi implementati dai dipendenti interni;
- costi operativi, legati all'aggiornamento e alla manutenzione e all'esercizio del software; questi costi denominati "costi operativi" comprendono: formazione di personale IT e di end-users, supporto degli end-users nei problemi riscontrati nell'utilizzo della tecnologia, gestione della sicurezza informatica, nell'utilizzo degli spazi aziendali connesso all'utilizzo delle tecnologie (centri, server farm, etc.), consumi di energia (elettrica, riscaldamento, condizionamento), costi di utilizzo della banda necessaria, costi derivanti dai tempi di fermo macchina dovuti a malfunzionamenti o errori degli end-users;
- costi legati alla dismissione del sistema (smantellamento delle apparecchiature hardware, eliminazione dei cavi portanti delle reti WAN / LAN).

L'approccio TCO è basato sulla considerazione che il costo totale di utilizzo di una apparecchiatura IT non dipende solo dai costi di acquisto, ma anche dai tutti i costi che intervengono durante l'intera vita di esercizio dello strumento.

I costi connessi alla stessa formulazione del TCO, che rientrano allo stesso modo nel TCO sono abbastanza rilevanti, dunque tale metodo viene spesso utilizzato solo da grandi aziende, in cui l'IT rappresenta una variabile strategica rilevante.

Il TCO è una analisi statica dei costi di esercizio di una apparecchiatura, e pertanto non tiene conto di eventuali ritorni dovuti all'investimento in innovazione, dei risparmi operativi che è possibile conseguire, delle nuove mansioni create; dunque il TCO monitorando solamente i costi non può essere usato in fase di analisi strategica degli investimenti.

Il TCO, anche se nato nell'ambito dell'IT, trova applicazione in qualsiasi altro campo, laddove sia possibile registrare i costi di esercizio di una determinata funzione aziendale.

2.1 Elementi di criticità del TCO

Il TCO è un buon modo per misurare i costi dei sistemi - ma non per stimarne il valore aziendale complessivo. Sfortunatamente, troppi siti permettono senza discutere che il TCO sostituisca un'analisi ben fatta del valore aziendale, come input alle decisioni di priorità sugli investimenti IT. **Se anche voi state lanciando, involontariamente, uno di questi boomerang decisionali vale la pena di dare un'occhiata alla reale natura del TCO e al ruolo che dovrebbe giocare nelle decisioni sull'IT.**

A prima vista, il total cost of ownership è un ottimo strumento d'analisi, un modo abbastanza semplice e lineare per capire e cominciare ad avere un miglior controllo dei veri costi dell'IT, e degli investimenti IT fra cui scegliere. **Ma sotto la pelle di rispettabilità contabile, c'è un processo che è facile manipolare. Non perché il concetto di TCO abbia qualche pecca fondamentale. Il problema è che il TCO è diventato così popolare che le sue analisi orientate ai costi diventano facilmente un sostituto per le ben più importanti (ma più difficili da calcolare) analisi del reale valore aziendale.** Quest'ultimo concetto è fondamentale, se vogliamo valutare correttamente le priorità degli investimenti.

2.2 Efficienza ed efficacia

Il TCO è solo una fetta della torta del valore, perché indirizza soltanto uno dei vari blocchi funzionali che compongono quello che si può definire il Full Business Value (FBV). **Il Full Business Value, in sostanza l'intero valore proprio di un investimento aziendale, ha quattro componenti: efficienza dei sistemi, efficacia dei sistemi, efficienza del business ed efficacia del business. Il TCO oggi si pratica prevalentemente ponendo l'enfasi sull'efficienza dei sistemi. E questo significa che il TCO copre solo un quarto di tutto lo scenario del valore.**

Il TCO è diventato popolare perché porta in evidenza abilmente dei costi effettivamente di natura IT, che sono rimasti trascurati per decenni. Per esempio, secondo Gartner, il costo completo di un PC in tutta la sua vita può essere di oltre cinque volte il suo costo d'acquisizione. Quest'affermazione illuminante è basata su un'attenta considerazione dei costi completi che richiede non solo mettere in esercizio il PC, ma supportarlo e mantenerlo durante il suo ciclo di vita. Man mano che questo concetto si diffondeva, i responsabili di progetto e di business case che avevano usato il TCO per valutare i costi totali erano coperti di elogi. Specialmente i direttori finanziari erano felici che il TCO identificasse queste spese, in termini chiari e facilmente misurabili. **La forza trainante del TCO crebbe sotto la spinta di commentatori che lo esaltavano in termini lirici. Invece di accettare la sfida di valorizzare altri tipi di benefici più difficili da misurare e più controversi - processi di qualità più alta, time-to-market più veloce, clienti più soddisfatti, dipendenti più sereni - molti capi dell'area finanziaria e dell'IT aziendale si arroccarono in un rigido abbraccio del TCO e cominciarono ad usarlo come giustificazione universale per gli investimenti IT.**

2.3 Ma il valore è un'altra cosa...

Ecco invece un esempio di come il TCO può rilevare importanti benefici, ma fallire nel descrivere il valore totale. **Supponiamo di voler motivare una richiesta di upgrade per un**

server e una rete, in una situazione di forte pressione perché l'IT riduca le sue spese. **Usando il metodo TCO di 'efficienza del sistema' catturiamo tutti i costi del ciclo di vita** per acquisto, installazione, gestione, supporto e manutenzione di quanto stiamo chiedendo. **A questo punto, se la scelta A** (mantenere la situazione attuale) **costa praticamente zero per l'acquisto** (c'è già), **ma costa molto di più per l'esercizio rispetto alla scelta B con i nuovi componenti**, l'analisi evidenzierà quantitativamente questi fattori. Così pure il TCO catturerà elementi come il costo e il rischio di interruzioni del servizio, i costi di manutenzione e di supporto. **Fin qui, bene: abbiamo un'analisi dei costi difendibile.**

Ma qual è il valore aziendale totale della nostra richiesta, per la nostra ditta? Per scoprirlo, dobbiamo aggiungere alla nostra analisi tre componenti non-TCO del Full Business Value. Per l'aspetto 'efficacia dei sistemi' potremmo quantificare fattori come la capacità di portare in esercizio applicazioni interamente nuove, per esempio il CRM, perché i nuovi server e rete avrebbero la capacità e le funzionalità necessarie. Per la parte di valore collegata all'aspetto 'efficienza del business' potremmo includere il valore di una riduzione sulle spese di viaggi a causa del miglior supporto fornito dal CRM, da un server potenziato e da un'infrastruttura di rete. Per il quarto aspetto, 'efficacia del business', potremmo considerare i valori tangibili e intangibili dell'avere un settore vendite in grado di introdurre più nuovi prodotti più rapidamente, sempre a seguito della introduzione del CRM.

Se usiamo il Full Business Value - non semplicemente il TCO - per giustificare gli investimenti IT, non solo potremo scoprire benefici aggiuntivi a supporto del nostro business case, ma aumenteremo la credibilità dell'IT dimostrando ai dirigenti degli altri settori la nostra conoscenza di tutta la macchina aziendale. Il problema è che il total cost of ownership è diventato così popolare che la sua analisi di costi sta diventando un sostituto per l'analisi completa di valore, più importante ma più difficile da eseguire.

2.4 Tre consigli per fare ordine

Se c'è il sospetto che il TCO possa essere strumentalizzato in un'organizzazione, questi sono tre consigli che possono aiutare a rimettere in ordine l'analisi dei valori.

Scoprire. Attenzione ai segnali di eccessiva dipendenza da TCO. Ad esempio: le metodologie TCO pesano per oltre il 25 per cento nelle analisi di valore e nei business case; quando si discute di analisi del valore si parla in termini di TCO e non di valore aziendale; i termini TCO e ROI sono usati come se fossero sinonimi; nei business case non si fanno valutazioni delle entrate e degli intangibili.

Trovare supporto. Se pensate che esista un problema di eccessiva concentrazione sul TCO, lanciate una campagna di aggiornamento dei valori culturali aziendali, con il supporto di altri dirigenti ed opinion makers. Chiedete loro di aiutarvi a convincere i titolari del processo di decisione sugli investimenti che il TCO non è tutto. Ricordate agli incerti che un sistema di decisione sugli investimenti trasparente, difendibile e robusto è il fondamento di una buona gestione dell'IT e dell'azienda.

Incorporare. Una volta trovato il supporto politico per questa rivoluzione dell'analisi dei valori, stabilizzate ed incorporate la visione FBV nel processo decisionale per gli investimenti IT aziendali. Aiutate il vostro comitato di selezione investimenti a inserire riferimenti FBV nei loro

requirements per la presentazione di business case. Sviluppate modelli di business case con relativa documentazione, chiara, che prescriva gli input ed indichi come ottenerli. Impostate un addestramento sull'analisi del valore, ricco di esempi buoni e cattivi. Per finire, date visibilità ai primi successi e siate sempre in cerca di modi per migliorare il processo man mano che progredite.

Il TCO può portare alla luce importanti benefici, ma non il valore totale. Come per altre buone idee che hanno in sé il potenziale per un cattivo uso, imparare ad usare il TCO in modo appropriato è un passo importante verso il successo.

3 UML: Introduzione

Questo articolo vuole essere una introduzione ai concetti racchiusi e presentati dalla modellazione UML. Quando si comincia a parlare di metodologie di [analisi](#) dei sistemi [software](#), spesso ci si incammina in territori quasi integralistici. L'obiettivo che mi sono preposto non è quello di spendere parole a favore o contro una particolare metodologia, quanto piuttosto quello di introdurre ai vantaggi dell'uso di un linguaggio ormai comunemente accettato (universalmente forse è una parola grossa) e che cerca di superare i vari problemi di cui soffrono, chi più chi meno, le diverse metodologie.

3.1 L'Analisi ed i Programmatori: un'incompatibilità atavica

UML è l'acronimo di Unified Model Language, e da come ci fa capire il nome rappresenta un linguaggio unificato per la modellazione. Non cadiamo nella trappola tesaci dalla parola linguaggio. Non si tratta di "linguaggio" nel senso che comunemente chi si occupa di programmazione attribuisce alla parola. Con l'UML infatti non si scrivono programmi, linee di codice, ma si descrive un "modello" di una situazione reale che deve essere successivamente codificato mediante un linguaggio di programmazione.

In altre parole, se mi trovo ad affrontare il problema della gestione di un carrello per l'e-commerce relativo ad un portale di componenti elettronici, tramite UML non preparo il codice relativo alla gestione del carrello, ma analizzo il dominio del problema e definisco "*cosa*" può significare "*gestire un carrello per l'e-commerce*".

Giunti a questo punto, come sempre avviene nella mente di chi si avvicina al mondo della modellazione, sorge la riflessione: "va bene, ma io il carrello in due minuti me lo scrivo, costruisco il db, preparo le pagine... 'sta modellazione mi fa perdere tempo...".

3.2 Non è vero.

In realtà, esistono due modi di scrivere codice. Il primo è scriverlo e basta; l'importante è che funzioni. Nel momento in cui dovesse presentarsi l'occasione di apportare delle modifiche, in quel momento ci si preoccuperà della cosa. Il secondo è quello di descrivere il problema, cominciando col porsi delle domande su "*cosa*" si vuole e, solo dopo aver chiaro in mente tutto questo, passare a considerare il "*come*" raggiungere l'obiettivo. Di primo acchito il secondo metodo è più dispendioso in termini di tempo. Alla lunga, invece, risulta essere il più remunerativo. Infatti un modello rappresenta una semplificazione della realtà.

Modellare quindi:

- **chiarisce bene le idee** - non solo le nostre ma anche quelle del cliente - quindi, ci permette di visualizzare "com'è" o "come vorremmo che fosse" un sistema;
- ci permette di **definire la struttura di un sistema**;
- ci permette di poter **suddividere i compiti di sviluppo** tra più persone, dato che ciò che deve essere fatto è ormai nero su bianco;

- ci lascia, sin dalle primissime fasi, una **traccia documentata delle attività** e delle decisioni prese;

Spezzata una lancia a favore dell'arte della modellazione, diamo anche una speranza a chi invece vede tutto questo come qualcosa di molto accademico ma poco attuabile nella realtà. Non sempre tutto va modellato (*Urrah!*). Ci sono delle situazioni in cui effettivamente si fa prima a scrivere il codice che risolve un problema che a descriverlo (*Doppio Urrah!*). Tuttavia io sono dell'idea che un minimo di analisi vada fatto sempre e comunque, proprio per evitare che presi dall'enfasi delle digitazione delle istruzioni si finisca per scrivere qualcosa che copre un problema e ne apre altri due.

Qualche riga fa ho usato il termine "analisi". Secondo me l'analisi non è altro che un sinonimo per "modellazione". La differenza, forse sottile, nei due termini sta nel fatto che mentre per fare analisi si possono seguire una pletera di metodologie, tutte quante con i loro fautori e detrattori, per fare modellazione oggi esiste un linguaggio universale che consente di creare, appunto, "modelli" software.

3.3 Un po' di storia

Vediamo un po' di storia per capire come siamo arrivati all'UML. A partire dagli anni '70 si è cercato di adottare un certo approccio ingegneristico alla scrittura di manufatti software. Possiamo riassumere questi tentativi fissando tre momenti storici:

- **anni '70:** Approccio Strutturato;
- **anni '80:** Information Engineering;
- **anni '90:** Approccio Object Oriented;

Durante questi tre momenti storici sono state espresse alcune linee guida che hanno ispirato poi la nascita di una o più metodologie e che ricoprivano diverse fasi del ciclo di vita del software.

L'**Approccio Strutturato** è stato il primo tentativo di introdurre delle regole nell'attività di produzione del software. Il suo concetto base era la *modularizzazione* e presentava una serie di diagrammi e tecniche specifiche per ognuna delle fasi dello sviluppo. Purtroppo risentiva della sua derivazione da ambienti di sviluppo basati su processi batch e, dato che richiedeva tecniche e modelli diversi per l'analisi ed il disegno, vi era una certa difficoltà nel passaggio tra una fase e l'altra.

L'**Information Engineering** ha fatto dei dati il suo punto di forza. Introduceva un uso estensivo di modelli formali e diagrammatici ed ha portato al fiorire degli strumenti *CASE*. Purtroppo quello che rappresentava il suo punto di forza si è dimostrato anche il suo "tallone d'Achille". Tuttavia il concetto di suddivisione in macrofasi, l'introduzione di strumenti automatici nel processo di sviluppo/manutenzione e la richiesta della partecipazione del cliente a tutte le fasi del ciclo, sempre di sviluppo/manutenzione, del software sono concetti che l'IT ha fatto propri.

L'**Approccio Object Oriented** supera la distinzione tra dati e processi e pone l'accento sulla *riutilizzabilità* dei componenti software. Il concetto di programmazione ad Oggetti porta ad una esplosione di metodologie che solo superficialmente si dichiaravano diverse, avendo come conseguenza una certa confusione negli strumenti CASE utilizzati per la progettazione OO.

A questo punto si comincia a sentire la necessità di disporre di un linguaggio ed una notazione universale per la creazione dei modelli software. Possiamo datare all'Ottobre 1994 la nascita di **UML**, quando Grady Booch e Jim Rumbaugh della Rational Software Corporation iniziano il lavoro di unificazione dei rispettivi metodi (Booch e OMT); Successivamente, Ivar Jacobson si unisce a questo processo di unificazione integrando anche il suo OOSE (Object Oriented Software Engineering). Il linguaggio che sta nascendo trae i suoi punti salienti da quelli delle metodologie che i suoi autori portano in dote. **OOSE** è orientato ai casi d'uso, il che fornisce un eccellente supporto all'analisi dei requisiti; **OMT** (Object Modeling Technique) è espressivo per l'analisi e adatto a sistemi data-intensive; **Booch** è particolarmente espressivo durante il progetto e adatto ad applicazioni engineering-intensive.

Nel gennaio del 1997, in risposta alla richiesta da parte dell'OMG (Object Management Group) della proposta di uno standard per l'Analisi ed il Disegno OO, **UML 1.0** viene sottoposto ad approvazione. Nel novembre 1997 UML diventa finalmente uno standard OMG.

3.4 Primo contatto con l'UML

Entriamo nel vivo di questa presentazione dell'UML. Da cosa è composto questo linguaggio? Possiamo pensare all'UML come all'insieme di tre elementi fondamentali: le *Viste*, i *Diagrammi* e gli *Elementi del Modello*.

Le **Viste** ci permettono di "guardare" il sistema da diversi punti di vista (funzionale, organizzativo, implementativo, ecc.). Mediante i **Diagrammi** è possibile descrivere le Viste. Mentre gli **Elementi del Modello** sono elementi stessi dei vari diagrammi e che prendono il nome di Attori, Classi, Packages, ecc.

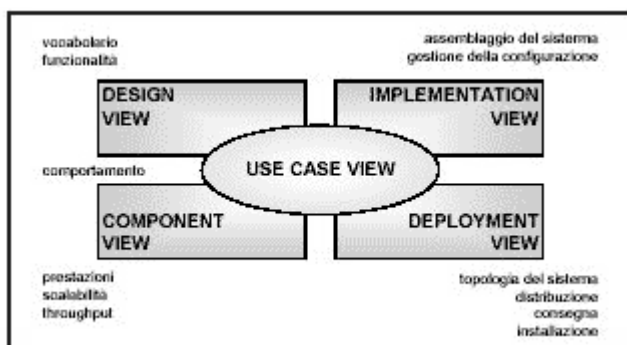


Figura 1 - Le Viste dell'UML

Facendo riferimento alla figura precedente possiamo vedere l'organizzazione in viste dell'UML.

Le Viste in dettaglio

La Vista *Use Case* viene utilizzata per analizzare i requisiti utente. I casi d'uso furono inventati da Ivar Jacobson per descrivere l'interazione tra uno o più utenti (attori) ed il sistema, mettendo

in evidenza le funzionalità del sistema da realizzare così come sono percepite dall'utente. Mediante gli Use Case è possibile definire il "cosa" del sistema. Discutere i casi d'uso con il cliente porta ad una completa comprensione delle funzionalità del sistema stesso.

Il "come" il Sistema debba funzionare viene descritto dalla vista **Design View**. Questa vista si compone di una parte statica rappresentata mediante i diagrammi strutturali (class ed object diagrams) e di una parte dinamica rappresentata mediante i diagrammi di comportamento.

La **Implementation View** permette di stabilire l'organizzazione e le dipendenze tra i diversi moduli in cui viene strutturato il codice.

La **Component View** rientra nell'analisi degli aspetti non funzionali del sistema e consiste nell'individuazione dei processi e dei processori, in modo da utilizzare in maniera più efficiente le risorse.

La **Deployment View** evidenzia l'architettura fisica del sistema e mostra come sono configurate ed allocate le unità hardware e software del sistema.

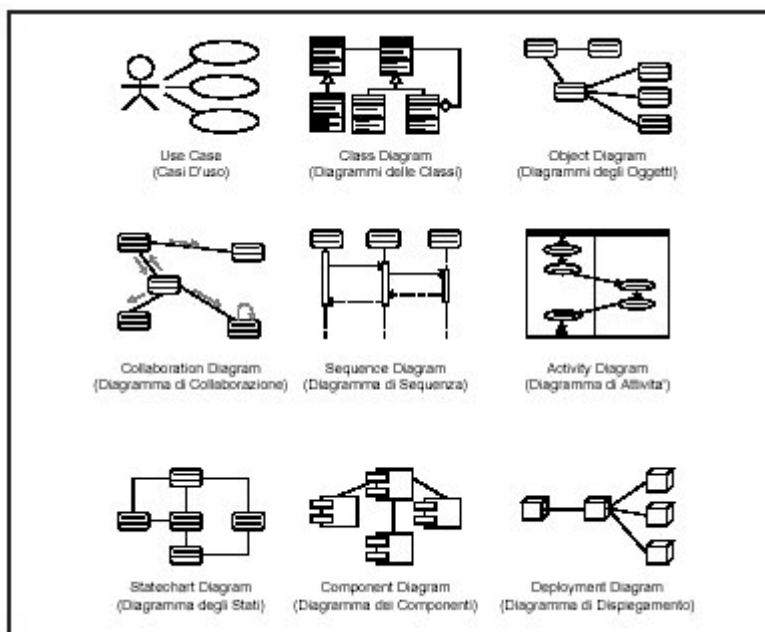


Figura 2 - I Diagrammi UML

3.5 Gli strumenti descrittivi

I diagrammi di cui dispone l'UML per descrivere le informazioni contenute nelle viste possono essere suddivisi nelle seguenti categorie:

- **Strutturali**
 - *Use Case Diagram*: mostra le modalità di utilizzo del sistema (casi d'uso), gli utilizzatori e coloro che interagiscono con il sistema (attori), le relazioni tra attori

e casi d'uso. Un caso d'uso rappresenta un possibile "modo" di utilizzo del sistema e descrive l'interazione tra attori e sistema, non la "logica interna" della funzione.

- *Class Diagram*: rappresenta le classi di oggetti del sistema con i loro attributi e operazioni. Mostra le relazioni tra le classi (associazioni, aggregazioni e gerarchie di specializzazione/generalizzazione).
- *Object Diagram*: sono una variante dei Class Diagram in cui le relazioni vengono dettagliate.

- **Di comportamento**
 - *StateChart Diagram*: è normalmente utilizzato per modellare il ciclo di vita degli oggetti di una singola classe. Mostra gli eventi che causano la transizione da uno stato all'altro e le azioni eseguite a fronte di un determinato evento.
 - *Activity Diagram*: rappresenta sistemi di workflow, oppure la logica interna di un processo (processo di business o processo di dettaglio), di un caso d'uso o di una specifica operazione di una classe. Permette di modellare processi paralleli e la loro sincronizzazione.

- **Di interazione**
 - *Sequence Diagram*: è utilizzato per definire la logica di uno scenario (specifica sequenza di eventi) di un caso d'uso (in analisi e poi ad un maggior livello di dettaglio in disegno). E' uno dei principali input per l'implementazione dello scenario. Mostra gli oggetti coinvolti specificando la sequenza temporale dei messaggi che gli oggetti si scambiano.

 - *Collaboration Diagram*: sono simili ai Sequence Diagram con la sola differenza che focalizzano sull'organizzazione degli oggetti che si scambiano i messaggi.

- **Di implementazione**
 - *Component Diagram*: evidenzia l'organizzazione e le dipendenze tra i componenti software.
 - *Deployment Diagram*: evidenzia la configurazione dei nodi elaborativi in ambiente di esecuzione (run-time), e dei componenti, processi ed oggetti allocati su questi nodi.

Perché scegliere l'UML

A questo punto sarete curiosi di sapere come utilizzare le viste ed i diagrammi descritti nei paragrafi precedenti. Se lo facessi soddisferei sicuramente la vostra curiosità, ma "sforerei" la lunghezza di questa introduzione. Per farli apprezzare non basta un piccolo esempio di *Use Case* o di *Sequence Diagram*. L'idea sarebbe quella di applicarli ad un **ipotetico caso di studio**. Quello che adesso vi propongo è allora una considerazione sui vantaggi e gli svantaggi dell'utilizzo di UML.

Partiamo dai **vantaggi**. UML mette fine alla "guerra delle metodologie" ponendosi come standard che assorbe i vantaggi di ciascuno dei vari metodi. UML focalizza l'attenzione sul processo di lavoro e non sulle tecnologie. In questo modo, l'aver un meta-modello comune, favorisce la comunicazione tra strumenti di supporto alla progettazione diversi e soprattutto tra i

diversi ambienti utilizzabili dai progettisti. Infine UML ha la capacità di adattarsi ad ogni tipologia di progetto, grazie alla sua complessa articolazione.

Ed arriviamo agli **svantaggi**. UML è complesso, inutile negarlo. D'altronde la complessità deriva dall'ambizione di poter rappresentare qualsiasi situazione possibile. Il numero dei suoi diagrammi è elevato ed alcuni di essi hanno piccole differenze che rendono difficile la scelta da parte del progettista. Vi è però una cosa da sottolineare. UML non va considerato in blocco. Non deve essere utilizzato per intero nell'economia di un sistema. I effetti, non necessariamente devono essere usati tutti i diagrammi di cui si dispone.

Questo dipende molto dalla sensibilità del progettista e può variare da sistema a sistema, ma, per esempio, a me piacciono molto gli *Use Case* per descrivere il sistema. Mi piacciono gli *Activity Diagrams* che permettono di descrivere le attività in cui scompongo il sistema. Mi piacciono i *Sequence Diagrams* per descrivere l'interazione temporale degli oggetti che implementano le attività in cui ho scomposto il sistema. Ad esempio, non sempre uso gli *StateChart Diagrams*. Lo faccio solo se voglio dettagliare una particolare sequenza di transizione.

Quindi, in poche parole, si vede che benché UML sia complesso è nello stesso tempo molto *adattabile e flessibile*. Anzi, proprio perché ogni progettista ha le sue esigenze, vi è una sorta di motivazione ulteriore ad adattare e ritagliare la metodologia sulle proprie esigenze o su quelle del progetto a cui si lavora.

3.6 Conclusioni

A questo punto, ci possiamo trovare di fronte a due situazioni possibili: vi ho annoiato oppure vi ho incuriosito. Nel primo caso, ehm, capita... ;-) Nel secondo, fatevi sentire! Il vostro numero potrebbe essere decisivo per determinare la nascita di uno o più articoli in cui affrontare l'applicazione della metodologia ad un caso di studio.

3.7 UML: Use Case

Dopo l'introduzione ad UML della prima parte di questa serie, affrontiamo l'applicazione di questa **tecnica di analisi e rappresentazione dei processi** applicandola ad un **caso reale**. Cioè, supponiamo di dover realizzare l'analisi di una problematica sottopostaci da un nostro cliente e descriviamola in modo da poter passare le specifiche al laboratorio dove si trovano i programmatori pronti a scriverne il codice. Nel farlo, cerchiamo di impostare il nostro ragionamento partendo dal presupposto di avere a che fare con una società di *modeste dimensioni* e limitate *risorse umane* da impiegare nel progetto, così da verificare nel contempo l'applicabilità di quanto diremo anche a questo tipo di realtà maggiormente diffuse.

Si potrebbe subito obiettare che questa situazione di suddivisione dei compiti, analisti da un lato e programmatori dall'altro si presenti solo nelle grandi società con diverse decine di persone, mentre se l'applicazione la dobbiamo scrivere da soli sarebbe più veloce realizzarla immediatamente, piuttosto che farne l'analisi.

Purtroppo l'esperienza insegna che nell'attività quotidiana le cose non sono così semplici. Una delle ragioni per cui sarebbe opportuno adottare un'analisi UML è che il cliente potrebbe - dato che ha sempre ragione! - cambiare idea sui requisiti dell'applicazione ed anche sul modo in cui debbano essere realizzati. Allora, perchè non "fissare dei paletti" prima di scrivere codice, piuttosto che andare a ritroso in quello che già abbiamo realizzato per adattarlo alle sue nuove esigenze? Questa operazione ci garantirà un miglior risultato (minor tempo di sviluppo e maggior qualità) e ci permetterà di chiarire subito i dettagli di implementazione, sia dal punto di vista del cliente che dal nostro, così come abbiamo già esaminato nella prima parte.

Bene. L'applicazione richiesta è un'**applicazione Web** per la **vendita di oggetti su Internet**. Infatti, il nostro cliente ci ha contattati dicendo: *"Vorrei vendere i manufatti che realizzo; vorrei un mercato non solo locale; mi piacerebbe che gli acquirenti potessero visionare un catalogo da cui scegliere; vorrei poter gestire gli ordini da qualunque posto perchè viaggio molto..."* e noi abbiamo risposto sicuri che quello che gli serve è proprio un'applicazione Web. La reazione classica del cliente è: *"Perfetto! Quando posso vedere qualcosa?"*.

A questo punto il povero programmatore si trova di fronte ad una **scelta** difficile: **1)** cominciare subito a scrivere l'applicazione, così tra due settimane potrà mostrare qualcosa di vivo e funzionante al cliente, oppure **2)** preparare un'analisi delle necessità indicate dal cliente così da poterle discutere con lui. Sebbene la prima strada può essere ritenuta la più affascinante, soprattutto per chi scrive codice, l'esperienza consiglia di adottare la seconda e adesso andiamo a vedere il perchè.

3.8 I requisiti utente

Per "requisiti utente" si intendono tutte quelle cose che al cliente piacerebbe poter fare con ciò che ci commissiona. Questi requisiti vanno analizzati in modo da poter separare le cose utili da quelle che non lo sono e da poter raccoglierne alcune in **single funzionalità**. A questo scopo, dopo che il cliente ci ha raccontato tutto quello che vorrebbe fare, il bravo analista si chiude nella sua stanza e comincia a pensare a come poter organizzare e dare forma alle idee ed ai *desiderata* del cliente.

Partiamo da questi: *"Vorrei avere la possibilità di creare un catalogo dei miei manufatti; vorrei un catalogo liberamente consultabile da chiunque; vorrei organizzare i manufatti raccogliendoli in serie; vorrei che gli interessati all'acquisto possano inviarmi un ordine, che io provvederò ad evadere previa una qualche forma di registrazione..."*.

Questi sono i *desiderata* del cliente. Vediamo come formalizzarli. Il primo passo è quello di ragionare sulle richieste e cercare di realizzare un **modello** di questa "realtà". Per far ciò utilizziamo, e cerchiamo di capire, gli **Use Case**. Innanzitutto, fissiamo quali possano essere le operazioni principali che si possono desumere dai requisiti:

- Si vuole gestire un **catalogo**, in cui gli oggetti possano essere organizzati in **categorie**.
- Il catalogo deve essere **liberamente consultabile** dagli eventuali clienti.
- Il cliente che decide di acquistare, deve **registrarsi** fornendo informazioni fondamentali quali: nome, cognome e indirizzo a cui spedire la merce.

A questi, aggiungiamo alcune cose che il cliente non ci ha esplicitamente chiesto, ma che implicitamente servono a far "girare" l'applicazione:

- Quando il cliente compila un ordine, si **notifica l'ordine** avvenuto.
- Una volta che l'ordine è stato evaso, si invia una **e-mail di conferma dell'evasione** al cliente.

Alla fine di questo processo, avremo disegnato qualcosa di simile alla **Figura 3**:

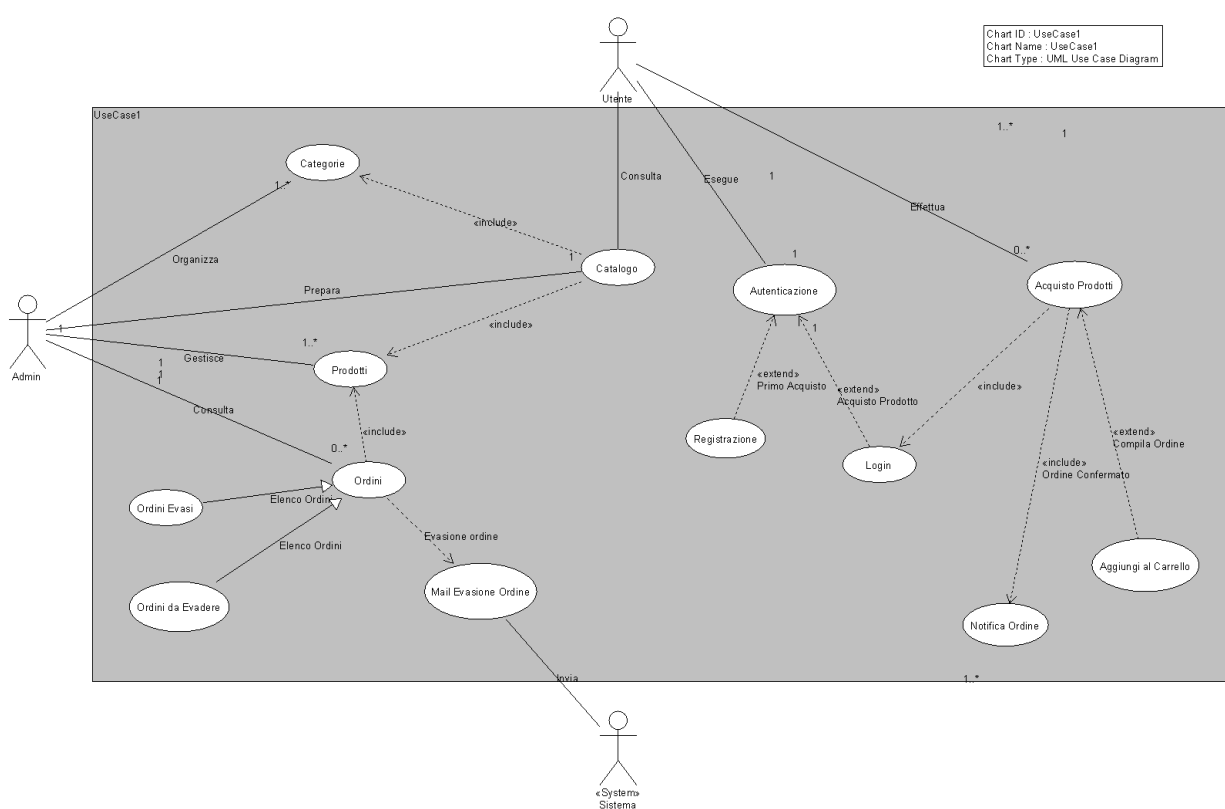


Figura 3

Elementi di uno Use Case

Facendo riferimento alla **Figura 3**, interpretiamo i vari elementi. Per prima cosa, fissiamo le idee sul significato dei simboli grafici. Uno **Use Case** ci fornisce fondamentalmente tre informazioni:

1. **chi** interagisce con l'applicazione,
2. **cosa** fa con essa,
3. **in che modo** lo fa.

Ovvero, in termini più formali ci rivela:

1. chi sono gli **attori** del sistema,
2. quali sono le **funzionalità** o servizi messi a disposizione dal sistema,
3. che **relazioni** esistono tra le funzionalità e tra queste e gli attori.

La **Figura 4** presenta questi elementi grafici. L'attore e lo **Use Case** sono facilmente riconoscibili.

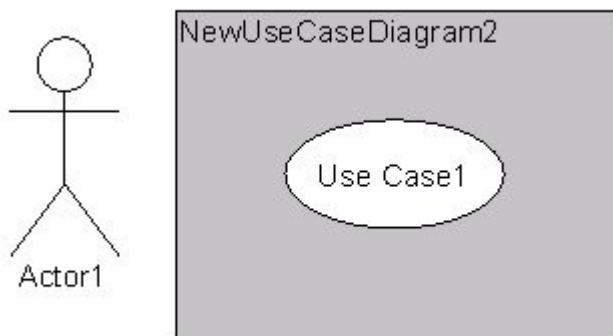


Figura 4

Per quanto riguarda le **relazioni**, possiamo averne di quattro tipi:

1. **Associazione:** è l'unico legame possibile tra attori e casi d'uso. Ci dice che un attore sta partecipando ad un caso d'uso, cioè interagisce con la funzionalità espressa dal caso d'uso.
2. **Generalizzazione:** la generalizzazione rappresenta una forma di specializzazione tra elementi dello **Use Case**. Ad esempio, se abbiamo un attore Olimpionico, due altri attori, Pugile e Ginnasta possono rappresentarne una specializzazione. Quindi ogni relazione in cui è coinvolto l'Olimpionico finirà per coinvolgere anche le categorie specializzate. Vale anche per gli **Use Case**.
3. **Estensione:** questa relazione è applicabile solo a casi d'uso ed indica il fatto che un'istanza di uno **Use Case** può includere, ma non necessariamente, nella sua funzionalità il comportamento di un altro **Use Case**. In altre parole il comportamento dello **Use Case** che riceve la freccia può includere quello da cui la freccia parte.
4. **Inclusione:** anche questa relazione ha senso solo tra **Use Case**. In questo caso l'oggetto da cui parte la freccia include nella sua funzionalità quella dell'oggetto che riceve la freccia. Necessariamente lo include.

3.9 Use Case ragionato

Dopo la teoria la pratica, cioè vediamo come nel nostro diagramma **Use Case** queste relazioni entrano in gioco. Nella **Figura 5** è indicata una relazione di associazione tra l'attore Admin e lo **Use Case** Prodotti.

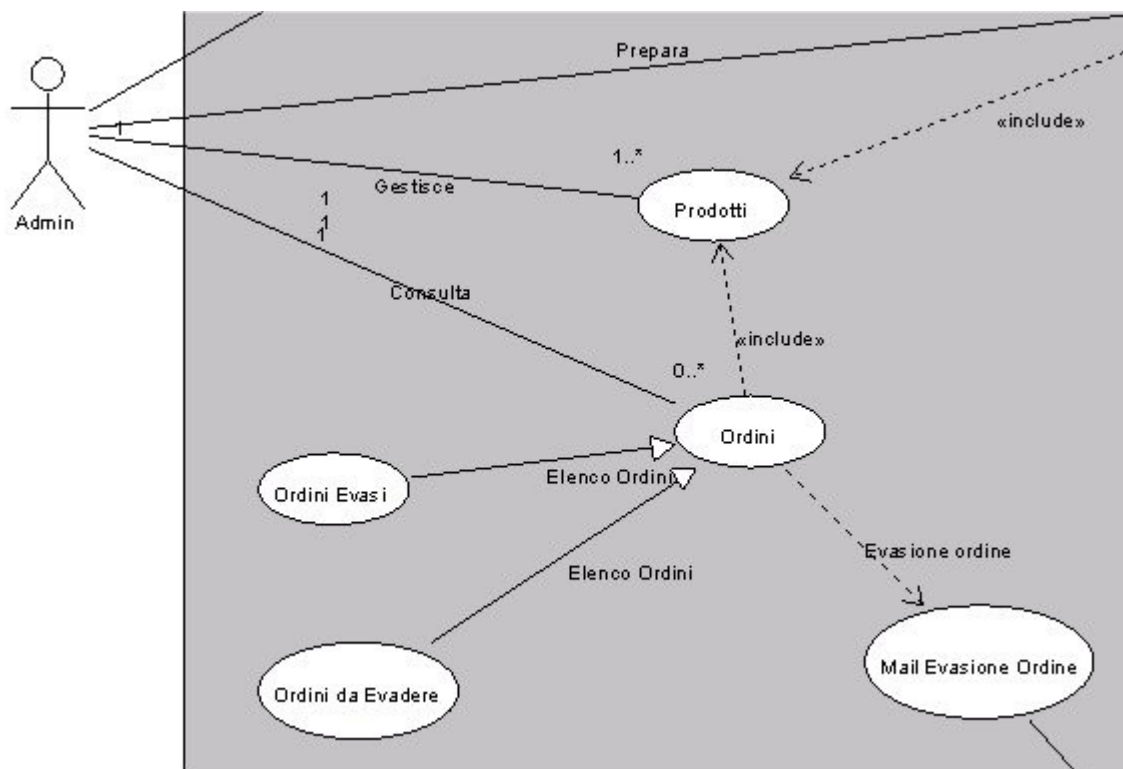


Figura 5

Questa associazione significa che l'amministratore del sito può gestire, come lascia intuire la stessa etichetta della relazione, una serie di prodotti. Infatti la molteplicità è **1..***, cioè uno a molti.

Spostando l'attenzione sugli **Use Case** vediamo che ci sono diverse relazioni. Soffermiamoci prima su quelle che coinvolgono lo **Use Case** Ordini. Come possiamo vedere sono due relazioni di inclusione che hanno come comprimari gli **Use Case** Prodotti e Mail Evasione Ordine. Come detto poc'anzi la relazione di inclusione richiede che nella funzionalità dell'oggetto da cui parte la freccia sia inclusa la funzionalità dello **Use Case** di arrivo della freccia.

Allora queste due relazioni cosa significano? Quella tra Ordini e Prodotti significa che un ordine contiene sempre dei prodotti, quindi almeno uno. La relazione tra Ordini e Mail Evasione Ordine presenta una informazione ulteriore. Non solo ci dice che per ogni ordine, ci deve essere una mail di conferma dell'ordine, ma in particolare la label sopra la relazione ci fa capire che la mail è necessariamente prodotta quando viene evaso un ordine.

Notiamo, infine, che sono specificate due specializzazioni dello **Use Case** Ordini, e cioè Ordini Evasi ed Ordini da Evadere. La label sulle due relazioni sottolinea il fatto che tutte le operazioni relative alla produzione di elenchi di ordini, che saranno applicate ad Ordini, saranno applicate anche alle sue due specializzazioni. Questa infatti è una relazione di **generalizzazione**.

Attenzione: non vale il contrario. Cioè, possono esserci alcune particolari funzionalità che Ordini Evasi implementa ma che non sono propogati ad Ordini.

Per commentare la relazione di **estensione** guardiamo la **Figura 6**:

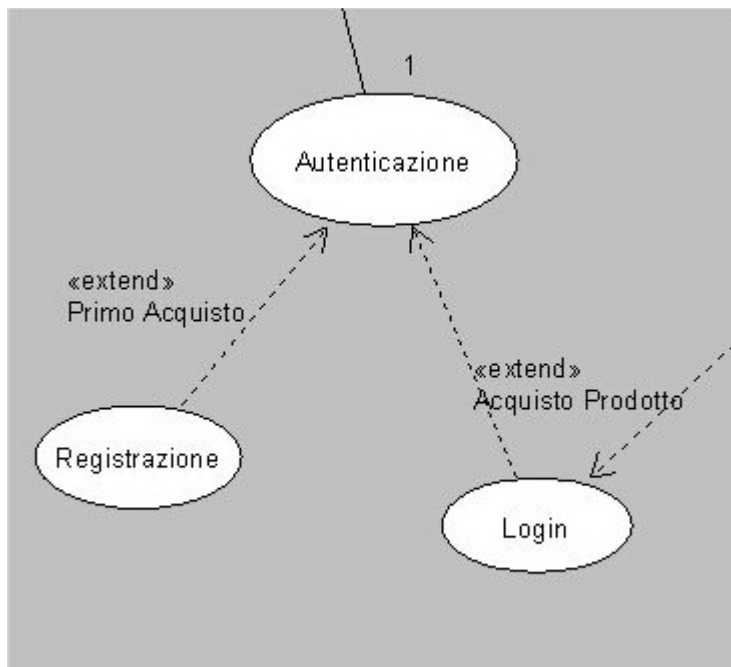


Figura 6

In essa vediamo due **Use Case** che estendono lo **Use Case** Autenticazione in due particolari situazioni. Nel caso di un primo acquisto si rende necessario effettuare una Registrazione, mentre nel caso di ogni altro acquisto successivo sarà sufficiente effettuare il Login. Come ricorderete, l'**estensione** non richiede necessariamente di includere la funzionalità di uno **Use Case** in un'altro. Allora possiamo usarla per esprimere un comportamento che deve essere perseguito in una particolare situazione, ma non sempre. In parole povere, non devo registrarmi ogni volta che voglio autenticarmi. La registrazione è richiesta solo nel caso del primo acquisto.

3.10 Conclusioni

Adesso che abbiamo il nostro **Use Case** (in un progetto reale ce ne saranno sicuramente più di uno) possiamo incontrarci con il nostro cliente per una prima validazione dell'analisi compiuta. Se il cliente riconosce le sue esigenze nella nostra rappresentazione, abbiamo completato il primo passo e cominciamo ad avvicinarci alla codifica.

Bibliografia

Hammer and Champy - Reengineering the Corporation

Davenport - Process Innovation

Riferimenti bibliografici per l'UML

- [Rational Rose](#)
- [UML Moka Book, Luca Vetti Tagliati](#)

Link

<http://www.devspy.com/public/Art/Misc/Art.aspx?area=tech&id=00025>

<http://www.devspy.com/public/Art/Misc/Art.aspx?area=tech&id=00036>