

Materiale Didattico

METODI E TECNICHE

CONSULENZA PER L'INNOVAZIONE
TECNOLOGICA
Redattore: Dr. Angelo SCARCIA

SERVIZIO FORMAZIONE

INDICE

	PAG.
I METODI E TECNICHE DI PROJECT MANAGEMENT	5
1 INTRODUZIONE	5
2 DEFINIZIONI	5
3 I PROCESSI DEL CICLO DI VITA	5
4 IL PROCESSO IMPOSTAZIONE	6
4.1 Il piano di progetto	6
4.2 Il piano di comunicazione	6
4.3 La Matrice delle aspettative	6
4.4 La Matrice di probabilità-impatto	7
5 IL PROCESSO PIANIFICAZIONE	7
5.1 La WBS	7
5.2 La OBS	8
5.3 Matrice delle Attività/Responsabilità	9
5.4 Utilizzo di tecniche reticolari	10
5.4.1 Il PERT	10
5.4.2 Il CPM	11
5.4.3 Il GANNT	12
5.5 Le risorse	13
5.6 I costi	13
5.7 Il piano di lavoro	14
6 IL PROCESSO AVVIO	14
7 IL PROCESSO CONDUZIONE	15
7.1 Consuntivazione	15
7.2 Controllo e riprogrammazione	15
7.3 Reporting	15
8 IL PROCESSO CHIUSURA E VALUTAZIONE	16

8.1	Chiusura del progetto	16
8.2	Valutazione del progetto	17
II	METODI E TECNICHE PER L'ANALISI DEI DATI E FUNZIONI	18
1	INTRODUZIONE	18
2	LE TECNICHE DEL VERSO DI PERCORRENZA DEL PROCESSO DI ANALISI	18
3	RAGNO FUNZIONALE	19
4	IL DFD (DATA FLOW DIAGRAM)	20
5	IL ERD (ENTITY RELATIONSHIP DIAGRAM)	22
6	IL PROTOTIPO (PROTOTYPING)	23
7	IL DOCUMENTO D'ANALISI	24
III	METODI E TECNICHE PER L'ANALISI TECNICA	25
1	INTRODUZIONE	25
2	IL MODELLO RELAZIONALE	25
3	DFD E STATI/EVENTI	27
4	IL PROTOTIPO (PROTOTYPING)	29
5	IL DOCUMENTO DI PROGETTO	29
IV	METODI E TECNICHE PER LO SVILUPPO	31
1	INTRODUZIONE	31
2	TECNICHE DI PROGRAMMAZIONE	31
2.1	Programmazione modulare	31
3	GLI ALGORITMI	32
3.1	Il Flow Chart (Diagramma a Blocchi)	33
3.2	La Pseudocodifica	37
3.3	Il linguaggio di programmazione	37
V	METODI E TECNICHE PER LA MISURAZIONE DEL PRODOTTO SOFTWARE	39
1	INTRODUZIONE	39

2 LE DEFINIZIONI	39
3 TIPOLOGIE DI METRICHE	40
4 METRICHE PER IL PRODOTTO SOFTWARE	41
4.1 Metriche dimensionali	41
4.1.1 LoC (Lines of Code - Linee di Codice)	41
4.1.1.11 DSI	42
4.1.1.21 Cocomo(COnstructive COst MOdel)	42
4.1.1.31 Cocomo II	42
4.1.2 Function Point	43
4.2 Metriche di Complessità Ciclomatica	43
4.3 Metriche per software Object Oriented	44
4.4 Metriche per la qualità del prodotto	44
VI METODI E TECNICHE PER IL TESTING ED IL COLLAUDO	46
1 INTRODUZIONE	46
2 IL TESTING	46
2.1 Principi generali	46
2.2 Difetti, errori e malfunzionamenti	46
2.3 Classificazione dei malfunzionamenti	46
2.4 Articolazione del processo di test	47
2.5 Le tipologie di tecniche	47
2.5.1 Tecniche di analisi statica	47
2.5.2 Tecniche di analisi dinamica	48
3 IL COLLAUDO	49

I METODI E TECNICHE DI PROJECT MANAGEMENT

1. INTRODUZIONE

I progetti con cui le aziende si confrontano diventano sempre più complessi per portata, impegno finanziario, grado di specializzazione delle competenze e pluralità di soggetti coinvolti: l'esigenza di rispettare gli impegni contrattuali assunti e di assicurare il rendimento dell'investimento aziendale impongono tuttora di utilizzare metodi accurati di previsione e controllo dei tempi e dei costi.

Il ricorso sempre più massiccio a queste tecniche è giustificato poiché:

- il tempo intercorrente tra l'inizio e la fine di un progetto tende ad allungarsi;
- i capitali impiegati in un progetto, prima che il risultato finale sia utilizzabile, tendono a crescere;
- l'impegno di tempo e denaro tende a divenire sempre più rigido al progredire del livello tecnologico;
- la tecnologia necessita di personale altamente specializzato;
- la specializzazione impone un'organizzazione del lavoro precisa;
- il maggior investimento di tempo e denaro, la rigidità dell'impegno, la necessità di organizzazioni altamente specializzate, ed i problemi che il mercato di questi fattori comporta esigono una pianificazione e un controllo efficienti.

I metodi e le tecniche di Project Management danno grossa enfasi alla corretta previsione dei tempi e dei costi e al controllo preciso degli avanzamenti per anticipare e risolvere gli effetti degli scostamenti.

2. DEFINIZIONI

Il Project Management, che da ora in poi indicheremo con l'acronimo PM, è un metodo di gestione sistemica di un'impresa complessa, unica e di durata determinata, rivolta al raggiungimento di un obiettivo chiaro e predefinito mediante un processo continuo di pianificazione e controllo di risorse differenziate, con vincoli interdipendenti di costi-tempi-qualità.

Il metodo utilizza tecniche di pianificazione e di gestione specifici, per coordinare situazioni complesse ed ottenere risultati positivi, per cui si propone come uno strumento di gestione che permette di:

- anticipare gli eventi, controllarli ed attuare azioni correttive;
- motivare le persone a raggiungere gli obiettivi.

Il Progetto, oggetto del PM è una combinazione di uomini, risorse e fattori organizzativi riuniti temporaneamente per raggiungere obiettivi unici, definiti, con vincoli di tempo, costo, qualità e numero di risorse.

3. I PROCESSI DEL CICLO DI VITA

Gestire un progetto durante tutto il suo ciclo di vita vuol dire mettere in atto una serie di processi gestionali fra loro fortemente relazionati.

Tali processi sono: Impostazione, Pianificazione, Avvio, Chiusura e Valutazione.

4. IL PROCESSO IMPOSTAZIONE

L'obiettivo di questo processo è definire i presupposti per garantire il raggiungimento degli obiettivi del progetto, nel rispetto dei vincoli di tempo e risorse (umane, economiche e tecnologiche) assegnate.

L'attività fondamentale del processo è la definizione del progetto che avviene mediante la creazione di:

- un piano di progetto quale guida di riferimento e di controllo da utilizzare nella conduzione del progetto;
- un piano di comunicazione e condivisione con gli attori coinvolti delle caratteristiche del progetto e delle responsabilità;
- una matrice delle aspettative;
- una matrice di probabilità-impatto per la valutazione del rischio.

4.1 Il piano di progetto

In particolare il piano di progetto deve contenere:

- Obiettivi
- Indicatori di successo
- Benefici attesi
- Organizzazione del progetto
- Descrizione del progetto
- Tempistica
- Stime impegno previsto
- Rischi.

4.2 Il piano di comunicazione

Il Piano di Comunicazione esplicita in maniera strutturata tutte le azioni necessarie per rispondere alle esigenze di comunicazione interne ed esterne che vengono ravvisate nel corso del progetto stesso.

Il Piano di Comunicazione va usato:

- durante la definizione del progetto;
- durante un'eventuale ridefinizione del progetto;
- ogni qualvolta vengano identificate nuove categorie di attori interni ed esterni al progetto;
- ogni qualvolta sorgano nuove esigenze di comunicazione verso gli attori coinvolti nel progetto.

4.3 La Matrice delle aspettative

Per aspettativa si intende il risultato che gli attori coinvolti nel progetto (Committente, Utenti, Direzione, etc.) si attendono dal raggiungimento degli obiettivi del progetto stesso.

Un'aspettativa deve essere espressa in modo chiaro ed in termini quantificabili, così da costituire un punto di riferimento e di confronto per il PM.

Obiettivo della Matrice delle aspettative è quello di raccogliere in modo sistematico ed esaustivo le aspettative di tutti i principali attori interessati al conseguimento degli obiettivi di progetto (Committente, Management, Utenti, etc.), al fine di:

- chiarire il più possibile cosa ci si deve (o meno) attendere dal progetto in termini di risultati e benefici;
- far emergere e risolvere eventuali discordanze rispetto agli obiettivi di progetto e tra le stesse aspettative raccolte;
- controllare, nel corso del progetto, l'allineamento tra i risultati prodotti e le esigenze espresse dal Committente e dagli Utenti.

4.4 La Matrice di probabilità-impatto

Una tecnica per analizzare il rischio collegato ad un evento è quella di formalizzare in una matrice la probabilità che l'evento accada, unita all'impatto che tale occorrenza potrebbe avere sul progetto finale, al fine di individuare in anticipo gli elementi che possono compromettere il raggiungimento degli obiettivi di progetto, con conseguente definizione delle azioni preventive e/o correttive più opportune.

La matrice va normalmente compilata prima della sezione riguardante i rischi nel Piano di Progetto.

L'importanza di queste attività consiste nel fatto che gestire un problema costa fino a 10 volte quello che serve a prevenirlo.

5. IL PROCESSO PIANIFICAZIONE

Il primo passo del processo di pianificazione è precisare: "Cosa fare?", segue poi la individuazione di "Chi fa?", tramite la descrizione della struttura organizzativa di progetto, quindi si fissa "Chi fa Cosa?" tramite la matrice della attività/responsabilità per giungere infine alla redazione del piano di lavoro utilizzando tecniche reticolari ed il diagramma di Gantt.

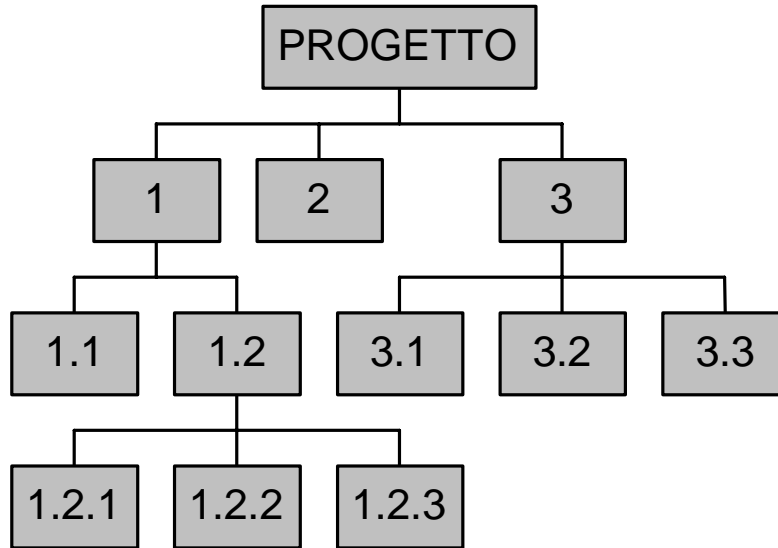
5.1 La WBS

Il PM propone una tecnica molto valida, e nel contempo semplicissima, per la gestione del "cosa fare" : la WBS.

La WBS - Work Breakdown Structure - è la scomposizione del progetto, o del lavoro, secondo un albero rovesciato, in cui partendo dalla radice (il progetto) e scomponendo il progetto in livelli di dettaglio successivi, dall'alto verso il basso, si arriva ad individuare le "foglie" dell'albero, generalmente note come work package o WP .

I WP corrispondono al livello ultimo di scomposizione prescelto in termini gestionali e, proprio per tale ragione, i WP devono poter essere controllabili e misurabili.

Per ogni WP occorre quindi specificare il nominativo di un unico responsabile e gli elementi che il WP realizzerà in output (deliverable) e che saranno oggetto di misura per il calcolo dell'avanzamento alla data. A titolo puramente esemplificativo si riporta il grafico generico di una WBS.



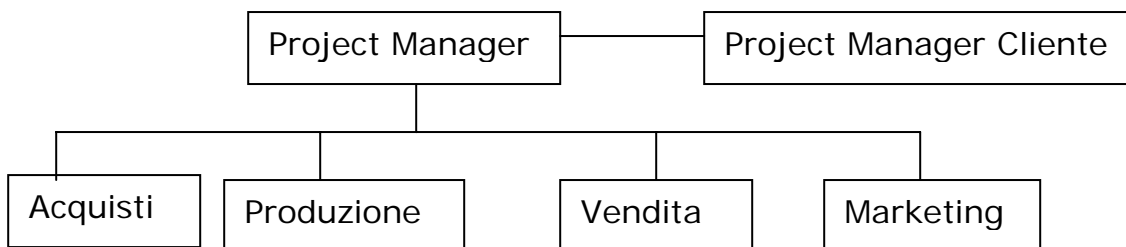
Completato il primo passo si passa quindi, attraverso il disegno di un'altra struttura ad albero, detta OBS, all'individuazione di: Chi fa?

5.2 La OBS

Con la OBS (Organization Breakdown Structure) si vengono ad individuare coloro che saranno, a tutti gli effetti, gli "attori" del progetto. L'OBS rappresenta la struttura organizzativa di progetto, la ripartizione strutturata dei livelli di responsabilità del progetto ed è uno strumento necessario a:

- identificare ed assegnare le responsabilità dei WP e delle attività che da esso verranno individuate;
- comunicare il flusso decisionale;
- integrare le informazioni tempi/costi secondo la struttura di responsabilità definita

A titolo puramente esemplificativo si riporta il grafico generico di una OBS.



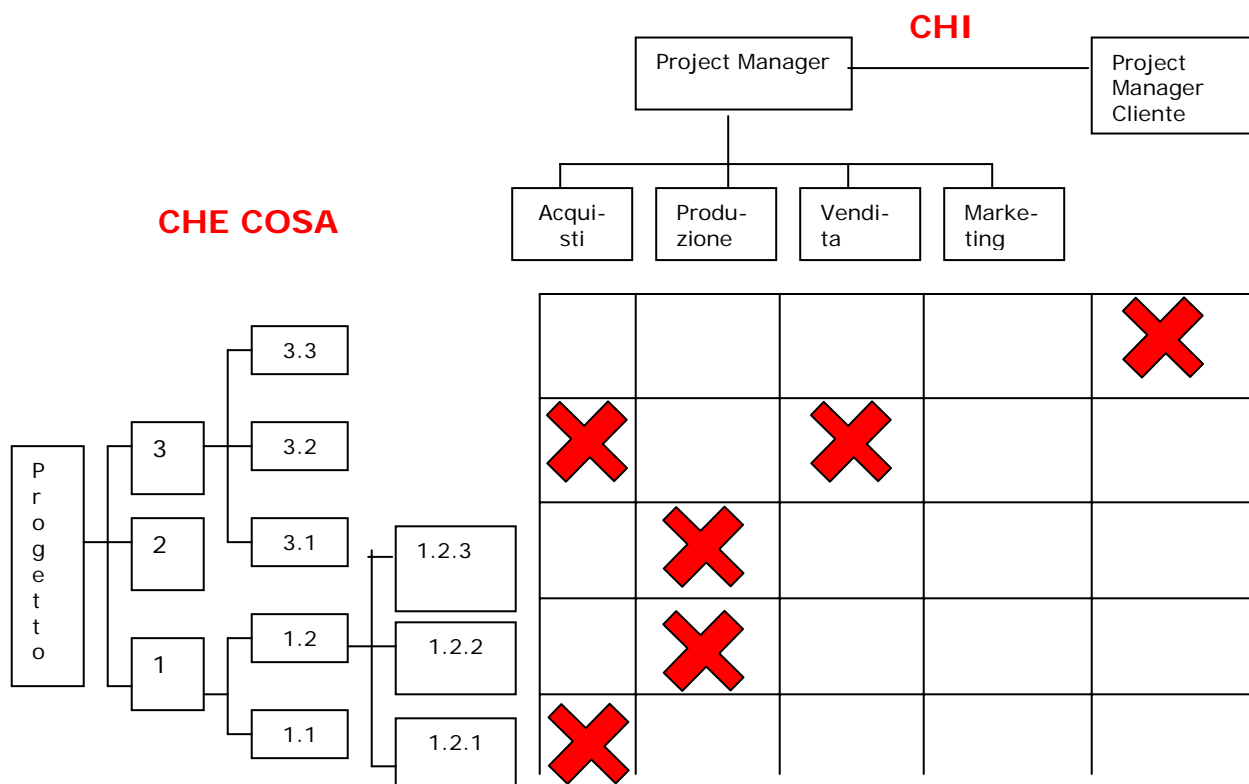
A questo punto occorre incrociare le due strutture WBS e OBS per definire una delle principali informazioni a supporto di una corretta gestione del progetto e per rispondere alla terza domanda: Chi fa cosa?

5.3 Matrice delle Attività/Responsabilità

Il grafico che risulta incrociando le strutture WBS e OBS, chiamato generalmente Matrice delle attività - responsabilità, indica per ogni WP l'ente, o meglio la persona, che dovranno ritenersi responsabili della realizzazione e rendicontazione del WP in oggetto.

A titolo puramente esemplificativo si riporta il grafico generico di una Matrice Attività/Responsabilità.

Matrice Attività – Responsabilità



5.4 Utilizzo di tecniche reticolari

In base alle tecniche precedentemente applicate, si ha a disposizione una lista di attività che rappresentano tutto il lavoro che deve essere svolto per portare a termine il progetto, con eventuali modifiche alla WBS rilevate a seguito della loro identificazione.

Per proseguire nella pianificazione si devono definire le varie dipendenze che sussistono tra un'attività e l'altra.

In genere sono definite quattro tipi di dipendenze:

1. fine-inizio secondo cui l'attività che segue non può iniziare se non è terminata la precedente;
2. inizio-inizio secondo cui l'attività che segue non può iniziare se non è iniziata quella che precede;
3. fine-fine secondo cui l'attività che segue non può terminare se non è terminata l'attività che precede;
4. inizio-fine secondo cui l'attività che segue non può terminare se non è iniziata l'attività che precede.

Definite le dipendenze deve essere valutata la durata delle singole attività sulla base del lavoro che ciascuna richiede; all'uopo si rende necessaria una stima della durata delle attività, completata con dati riguardanti:

- un possibile scarto temporale, in positivo ed in negativo, per ogni singola attività;
- l'identificazione di una probabilità di completamento dell'attività nel tempo stimato.

Con la definizione di questi dati, si può determinare un reticolo e quindi la durata e fine del progetto.

Le principali tecniche reticolari che vengono utilizzate durante la pianificazione del progetto sono:

Program Evaluation and Review Technique(PERT);

Critical Path Method(CPM) o analisi del cammino critico.

5.4.1 II PERT

La tecnica PERT consiste nell'ordinare graficamente varie attività che a causa della loro reciproca dipendenza e della loro successione cronologica, concorrono tutte al completamento del progetto

Uno dei primissimi passi è la costruzione del modello grafico, dove si esige un preliminare esame approfondito del progetto da portare a termine al fine di dividerlo in tante attività elementari per studiare al meglio ogni singola attività.

Ogni attività viene caratterizzata dalla sua durata, dal tipo di relazioni con le altre attività, dal costo e dalle risorse.

Il PERT è un approccio probabilistico alla stima della durata delle attività, utilizzato in particolare quando vi sia alta incertezza sulla disponibilità di risorse, sui fabbisogni e sulla resa delle risorse.

Il PERT ci fornisce:

- i legami logici fra le attività ;
- un aiuto per calcolare la durata totale di un progetto (o di un gruppo di attività);
- un aiuto per riconoscere i cammini critici;
- un aiuto nella revisione delle attività per accorciare la durata del progetto.

Il PERT permette di disegnare quello che potremmo chiamare il "modello" del progetto.

Tale modello è descrivibile rappresentando in forma grafica le attività del progetto e le relazioni esistenti fra le attività stesse.

Attraverso la rappresentazione di tale modello grafico vengono quindi individuati i vincoli logici, o se si preferisce, propedeutici, esistenti fra le attività, considerando le sequenze operative, i cicli di vita, che si intende seguire.

Il limite del PERT è che:

- trascura i cammini quasi critici;
- assume che le durate delle attività siano indipendenti;
- la stima della durata della attività può essere inadeguata;

La precisazione delle valutazioni della durata o/e del costo delle operazioni di un progetto è invece il presupposto per l'impiego di un'altra tecnica: il CPM (Critical Path Method)

5.4.2 II CPM

Il CPM parte dal presupposto che le esperienze fatte ci liberino dalla incertezza dei tempi, mentre rimane quella relativa ai costi in quanto l'obiettivo principale è quello di minimizzare il costo totale e su questa base si fissano i tempi di realizzazione.

Il CPM richiede:

1. una stima esatta del costo e del suo andamento al variare della durata dell'operazione in cui si riferisce;
2. una sicura determinazione dei limiti di tempo entro i quali l'operazione potrà effettuarsi.

Nel caso di impiego del CPM, il controllo e l'aggiornamento del reticolo hanno una importanza inferiore rispetto al PERT, essendo perfettamente determinati fin dall'inizio durate, costi, e altri parametri.

L'obiettivo principale del CPM è quello di ricercare le criticità in un percorso individuato su un grafo connesso ed orientato.

5.4.3 Il Gantt

Se il PERT ed il CPM rappresentano la relazione logica delle attività, il diagramma di Gantt rappresenta le attività su assi cartesiani con lo scorrere del tempo in ascisse e la successione delle attività in ordinate.

La rappresentazione permette di pianificare nel tempo qualsiasi successione di operazioni, di controllarne lo stato di avanzamento, con particolare riferimento a eventi o date chiavi (milestones), e di verificare il grado di completamento del progetto

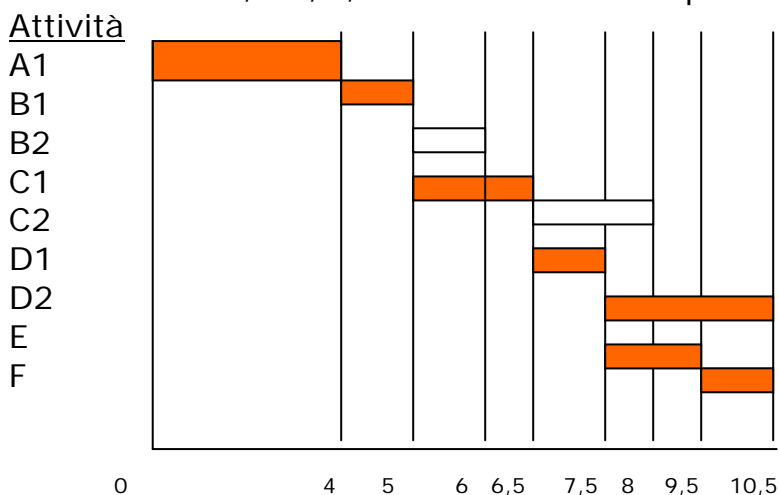
Con il diagramma di Gantt si vede quanto tempo occorre per giungere alla soluzione del problema o comunque alla realizzazione del progetto.

Il progetto generale può essere suddiviso in progetti secondari per facilitarne la gestione; in tal caso avremo un Gantt generale e altri Gantt secondari che faciliteranno il controllo particolareggiato dell'avanzamento dei progetti.

Il diagramma di Gantt è utile anche per confrontare i tempi previsti con i tempi realmente impiegati, in modo da valutare se la previsione era stata fatta bene.

Con il diagramma di Gantt è anche possibile evidenziare il cammino critico.

Rappresentiamo ora un tipico schema di diagramma di Gantt relativo alle attività A1, B1, ..., F di un determinato percorso:



dove le attività critiche sono evidenziate con fondo rosso e ciascuna attività è rappresentata attraverso un segmento posizionato, con riferimento alla scala temporale, alla data di inizio di quella attività e di lunghezza pari alla durata dell'attività stessa.

Poichè in un progetto alcune attività possono iniziare se si sono completate determinate attività, è necessario rispettare nel programma i vincoli di sequenza.

Se ad esempio un'attività può iniziare solo se le attività che la precedono sono completate, allora occorre posizionare la data di inizio di quella

attività subito dopo quella del completamento dell'ultima attività che la precede.

Tale tecnica risulta particolarmente complessa quando si hanno molte attività con vincoli di precedenza tra loro, infatti, i vincoli non possono essere facilmente evidenziati nel grafico.

5.5 Le risorse

Oltre a definire le attività di un progetto e le relazioni (propedeuticità) fra le stesse, è indispensabile un'analisi sulle esigenze del progetto dal punto di vista delle risorse. Per risorse si intendono generalmente tutti quei fattori produttivi, esprimibili attraverso quantità, che sono necessari per l'espletamento delle attività di progetto. Sono quindi da considerare risorse le persone, i materiali ed i macchinari che verranno impiegati. Una volta definita l'anagrafica delle risorse a disposizione (Pool di risorse), si assegnano alle singole attività del progetto le quantità di risorse necessarie. Sempre grazie agli algoritmi CPM è possibile schedare le risorse nel tempo, definendo i cosiddetti "carichi" di risorsa periodici che, espressi tramite istogrammi, facilitano l'individuazione delle risorse critiche e di eventuali sovra o sotto carichi (over-under load).

E' evidente come la presenza di eventuali sovraccarichi metta in forte dubbio la possibilità di realizzare il progetto rispetto all'attuale pianificazione temporale, e tutto ciò dovrebbero indurre il project manager a cercare possibili variazioni al grafico di Gantt che ottimizzino l'uso delle risorse (livellamento risorse) eliminando, per quanto possibile, eventuali picchi eccessivi.

5.6 I costi

Valorizzando i carichi di risorsa necessari attraverso appositi costi unitari (rateo) è possibile individuare la curva dei costi di progetto derivanti dalle risorse. Tale curva, nota come "Curva ad S" del budget, è solitamente riferita ai soli costi diretti derivanti dall'utilizzo di risorse sul progetto. In alcuni casi, per considerare anche le aliquote derivanti dai costi indiretti, il rateo di costo di una risorsa (es: analista senior) potrebbe già contenere in se il ribaltamento dei costi medi giornalieri derivanti dalla componente indiretta o, comunque, di carattere generale. L'assegnazione dei costi ad un progetto e la sua analisi può essere effettuata a vari livelli di dettaglio e organizzativi.

Per i progetti complessi è alquanto difficile gestire i costi di progetto a livello di singola attività, per cui risulta essere molto utile, in tal caso, la WBS di progetto nella definizione di massima del budget di progetto.

Valorizzando infatti, i soli work package finali della struttura è possibile calcolare costi aggregati ai livelli superiori.

In base alla programmazione temporale è possibile "fasare" i costi di progetto rispetto ad un calendario temporale, definendo, come già accennato la curva ad S dei costi pianificati, detta comunemente BCWS - Budget Cost o of Work Scheduled.

5.7 Il piano di lavoro

Il Gantt, gli istogrammi di carico delle risorse e la curva ad S dei costi, una volta integrati, ottimizzati ed ufficializzati, vanno a costituire il Piano di lavoro.

Tale documento originale costituisce la Baseline di progetto rispetto alla quale verranno espletate le attività di monitoraggio e controllo degli avanzamenti.

In tale documento in particolare vanno definiti:

- la successione delle milestones, ovvero gli obiettivi intermedi da perseguire in termini di tempo, costo e qualità;
- la responsabilità per il raggiungimento per ciascun milestone;
- le gerarchie, i processi decisionali ed i flussi informativi all'interno dell'organizzazione del progetto;
- le attività indispensabili per la piena realizzazione dei prodotti finiti associati a ciascuna milestone;
- l'assegnazione delle risorse e delle responsabilità per ciascuna attività;
- la successione delle attività e le precedenze fra le stesse per poter così programmare le date di realizzazione di ciascun obiettivo intermedio.

6. IL PROCESSO AVVIO

Il processo prevede che venga indetta una riunione di avvio (Kick off di Progetto).

Lo scopo della riunione di kick off di progetto è di notificare formalmente agli interessati al progetto che questi è iniziato, ed assicurarsi che tutti hanno realizzato i propri ruoli e responsabilità.

La riunione di kick off è l'occasione per riunire i membri del gruppo di lavoro, i responsabili del cliente, gli altri interessati e formalmente annunciare l'inizio del progetto, discutendo l'approccio generale del progetto, la sua tempificazione e dirimere tutti i possibili dubbi.

Il processo di avvio prevede inoltre la definizione della modulistica che verrà utilizzata.

7. IL PROCESSO CONDUZIONE

Una volta creato il piano di lavoro ed avviato il progetto, si passa al processo di conduzione, che si articola in tre fasi:

1. Consuntivazione;
2. Controllo e riprogrammazione;
3. Reporting.

Il processo si attua attraverso una continua ciclicità delle tre fasi poggiandosi su uno dei pilastri concettuali del PM: non esiste controllo se non esiste un piano di riferimento; non ci si deve però limitare ad una semplice rilevazione dei dati effettivi (consuntivo) del progetto ma poi, per cercare di individuare eventuali scostamenti, bisogna mettere a confronto il dato effettivo, l'avanzamento del progetto, con quanto precedentemente previsto a livello di piano.

Sarà proprio l'analisi di tali scostamenti e delle performance del progetto (tempi, costi e qualità) a permettere di elaborare le nuove stime a finire.

7.1 Consuntivazione

La consuntivazione ha come obiettivo la raccolta tempestiva di dati accurati.

Per ogni attività, ad una certa data N, vanno presi in considerazione i rapporti dei tempi individuali (Time Sheet) alla data N ed il Piano di Lavoro aggiornato alla data N-1.

L'elaborazione di tali documenti permettono di fornire:

- le spese sostenute alla data N;
- i dati di consuntivo del periodo corrente;
- le stime a finire dichiarate dai componenti del gruppo di progetto per il periodo successivo.

7.2 Controllo e riprogrammazione

La fase di controllo e riprogrammazione ha l'obiettivo di verificare il corretto avanzamento dei lavori rispetto al piano di lavoro ed individuare ed analizzare gli scostamenti, le cause ed effetti di tali problematiche, le eventuali azioni correttive ed il relativo impatto sul piano di lavoro.

Viene preso in considerazione il piano di lavoro aggiornato alla data N-1 ed i dati di consuntivo e vengono prodotti:

1. il piano di lavoro aggiornato alla data N in termini di consuntivo, stime a finire e costi;
2. il documento relativo allo stato di avanzamento lavori.

7.3 Reporting

L'obiettivo del Reporting è quello di dare informazioni sull'andamento del progetto.

La fase di reporting utilizza il piano di lavoro aggiornato alla data N e fornisce il documento denominato Stato Avanzamento Lavori(SAL) alla data N.

Il SAL ha l'obiettivo di fornire le informazioni di carattere quali/quantitative necessarie a governare l'andamento del progetto in corso, in termini di:

- rispetto dei tempi e delle scadenze;
- rispetto dei costi e degli impegni di risorse;
- valore aggiunto degli investimenti.

Nel SAL sono riportate tre tipi di informazioni chiave:

1. tempistica;
2. impegno risorse;
3. criticità.

Per quanto riguarda la tempistica sono fornite nel documento informazioni che permettono di avere una visione complessiva del rispetto della tempistica da parte del progetto, di eventuali scostamenti rispetto ai piani, delle eventuali implicazioni su altri progetti e dei margini di flessibilità disponibili.

Per l'impegno delle risorse sono riportate informazioni che permettono di avere un quadro complessivo dell'impegno delle risorse (umane ed economiche) e di evidenziare eventuali scostamenti rispetto ai piani.

Relativamente alle criticità sono fornite informazioni che permettono di avere una visione complessiva delle eccezioni e delle criticità emerse sul progetto e che non possono essere risolte a livello di progetto, ma richiedono una decisione/soluzione di più alto livello

8. IL PROCESSO CHIUSURA E VALUTAZIONE

La chiusura del progetto è costituita da due fasi:

1. Chiusura del progetto;
2. Valutazione il progetto.

8.1 Chiusura del progetto

L'obiettivo della fase Chiusura del progetto è quello di:

- formalizzare la conclusione del progetto e consegnare i risultati al committente;
- capitalizzare le competenze sviluppate nel corso del progetto in termini di contenuti, metodi, stime e impegno di risorse.

La fase utilizza l'ultimo Stato Avanzamento Lavori ed i risultati del progetto ed espleta le attività di chiusura fornendo:

- il rapporto finale con una disamina generale su tutta la vita del progetto;
- la scheda per la chiusura di progetto per porre l'attenzione su cosa il progetto doveva produrre e cosa il progetto ha realmente prodotto;
- azioni di comunicazione per la chiusura del progetto per il trasferimento delle conoscenze al team di supporto, il trasferimento dei documenti completi, il trasferimento della lista dei lavori pendenti, etc.;

- dati e conoscenze progettuali da registrare per il database delle stime per futuri progetti simili ;
- rilascio delle risorse assegnate al progetto.

Il progetto non viene considerato completato finché le attività di chiusura non sono state svolte.

8.2 Valutazione del progetto

L'obiettivo della fase Valutazione del progetto è quello di:

- valutare l'andamento ed i risultati del progetto in termini di:
 - rispondenza alle aspettative della committenza
 - rispondenza ai vincoli di durata ed impegno stimati
- valutare le risorse impegnate sul progetto in termini di:
 - performance dimostrate
 - crescita professionale
- riesaminare il progetto ed evidenziare il percorso critico (p.e. metodologia utilizzata, errori da non ripetere, etc)

La fase utilizza la Matrice delle Aspettative, il Piano di progetto ed il Piano di lavoro e fornisce la Scheda per la Valutazione finale di progetto che comprende una scheda informativa sull'andamento progettuale, una valutazione sulla soddisfazione degli attori, una valutazione sulle risorse, una validazione finale di progetto.

La fase di valutazione è necessaria per:

- valutare gli scostamenti(es. tra costi preventivi e costi finali);
- valutare i risultati ottenuti;
- individuare gli errori e le modifiche apportate;
- verificare le conseguenze degli errori;
- stabilire come potranno essere evitati nel futuro;
- evidenziare le necessarie modifiche di miglioramento.

II METODI E TECNICHE PER L'ANALISI DEI DATI E FUNZIONI

1. INTRODUZIONE

L'analisi dei dati e delle funzioni ha il compito di approfondire la conoscenza del dominio del problema in maniera tale da poter descrivere, in modo preciso e completo, il "cosa" un progetto deve realizzare indipendentemente dagli aspetti implementativi. Inoltre, essa deve definire in modo rigoroso e formale tutte le funzioni del sistema da automatizzare.

Gli obiettivi dell'analisi concettuale quindi sono:

- la *conoscenza approfondita* del dominio del problema, senza trascurare alcun dettaglio;
- il *consolidamento* della suddivisione del sistema nei sottosistemi individuati nella fase di analisi e definizione dei requisiti;
- l'elaborazione dello *schema concettuale delle funzioni* (Data Flow Diagram);
- l'elaborazione dello *schema concettuale dei dati* (diagramma E-R),
- un protocollo di comunicazione comune fra l'utente e l'analista (prototyping);
- la produzione del *documento di analisi* che, seppur derivato da quello dei requisiti, rappresenta comunque un prodotto, in quanto descrive la realtà d'interesse – sempre da un punto di vista concettuale – ma ad un livello di astrazione differente.

Allo scopo di aumentare la conoscenza del dominio del problema, la fase di analisi concettuale prevede due attività fondamentali:

- l'analisi concettuale delle funzioni.
- l'analisi concettuale dei dati,

Nel presente documento verranno analizzati i metodi e le tecniche utilizzati, nella fase di analisi concettuale, per la elaborazione dei dati e delle funzioni ai fini della conoscenza in dettaglio del dominio del problema.

2. LE TECNICHE DEL VERSO DI PERCORRENZA DEL PROCESSO DI ANALISI

Esistono due tecniche che caratterizzano il verso di percorrenza di un processo di analisi concettuale, sia per l'analisi delle funzioni che per l'analisi dei dati, la tecnica **top-down** e quella **bottom-up**.

La prima si avvale di trasformazioni che, applicate ad un singolo concetto, ne produce una descrizione più dettagliata; la seconda, invece fa uso di trasformazioni che introducono nuovi concetti e proprietà, non presenti nello schema iniziale.

Nella tecnica top-down lo schema concettuale viene prodotto mediante una serie di raffinamenti successivi, a partire da uno schema iniziale,

che descrive tutte le specifiche con pochi concetti molto astratti; lo schema viene via via raffinato mediante opportune trasformazioni, che aumentano il dettaglio dei concetti presenti.

Nella tecnica bottom-up, l'analisi parte dalle componenti più piccole, che descrivono frammenti elementari della realtà d'interesse. A questo punto le varie componenti vengono rappresentate da semplici schemi concettuali, che possono consistere anche in semplici concetti. I vari schemi così ottenuti vengono poi fusi fino a giungere, attraverso una completa integrazione di tutte le componenti, allo schema concettuale finale.

Ognuna delle tecniche citate presenta vantaggi e svantaggi.

La top-down ha il vantaggio di permettere una descrizione delle specifiche essenziale, trascurandone i dettagli, ma per fare questo è necessaria sin dall'inizio una visione globale ed astratta di tutte le componenti del sistema e questa, quando il sistema è complesso, è difficile da avere.

Al contrario, la strategia bottom-up si riesce ad adattare ad una decomposizione del problema in componenti più semplici, lo svantaggio risiede nel fatto che, a volte, l'integrazione delle diverse componenti risulta difficoltosa.

Esiste allora la possibilità di adottare una tecnica mista, con la quale si cerca di combinare i vantaggi della tecnica top down con quelli della bottom-up

3. RAGNO FUNZIONALE

Alla base del Ragno Funzionale vi è la rappresentazione della realtà tramite funzioni, cioè insieme di attività omogenee, che per essere attivate hanno bisogno di informazioni in ingresso ed il cui prodotto sono ancora informazioni.

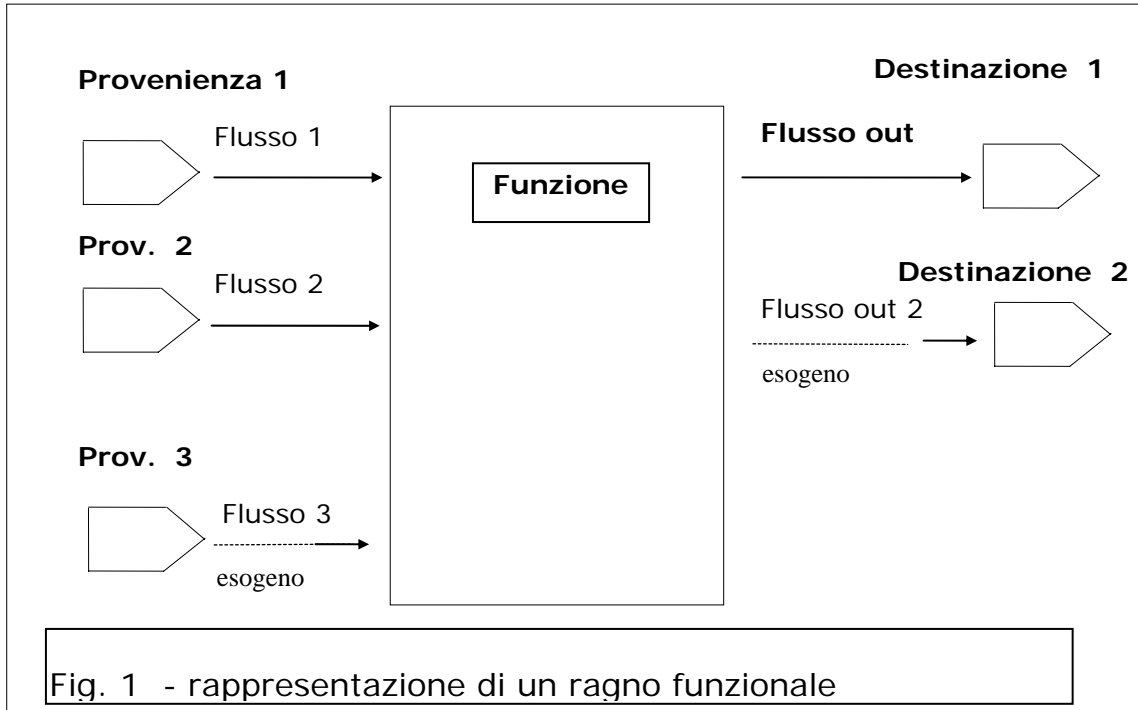
Il Ragno Funzionale è dunque la rappresentazione grafica di una funzione, definita con un rettangolo a cui si assegna un nome significativo, in cui vi sono in ingresso ed in uscita i flussi informatici, con l'indicazione della rispettiva provenienza e destinazione.

La simbologia rappresentata in Fig.1 è così formulata:

rettangolo, che rappresenta una funzione;

freccia, che rappresenta un flusso di informazioni omogeneo;

connettore, che indica la provenienza o la destinazione.



I flussi endogeni, cioè interni all'azienda, sono rappresentati con frecce a tratto continuo; i flussi esogeni, cioè i flussi che provengono dall'esterno o che vanno verso l'esterno, sono rappresentati con frecce tratteggiate. La forma di rappresentazione consente di focalizzare l'attenzione sulle interfacce della funzione in esame, e non come è articolata al suo interno, permettendo così di evidenziare i flussi informativi, e come essi attraversano le singole funzioni.

È possibile porre nello stesso diagramma le funzioni che si riferiscono ad una stessa area problema, connesse tra loro tramite i relativi flussi di ingresso uscita: questi diagrammi si indicano con il nome di "diagrammi di correlazione".

Occorre sottolineare i limiti di tale tecnica di rappresentazione, che consistono nel non evidenziare le basi di dati, che pur esistono nel contesto reale in analisi, ma i flussi informativi.

4. IL DFD (DATA FLOW DIAGRAM)

Il metodo più ampiamente riconosciuto per descrivere l'aspetto funzionale del dominio del problema è il Data Flow, il cui schema è il DFD.

La tecnica dei DFD fornisce un supporto completo per l'espletamento delle attività di analisi e di modellizzazione di un sistema informativo.

Questa tecnica è utilizzabile per analizzare con vari gradi di approfondimento un'area problema, che può essere composta sia da una realtà organizzativa che da applicazioni informatiche.

La tecnica consiste in una rappresentazione grafica che evidenzia i processi ed il flusso dei dati che tali processi utilizzano e/o trasformano.

Gli elementi del modello sono:

- Flusso di dati (Data flow)
- Processo
- Data store (Deposito)
- Agente esterno.

Il data flow è definito come un flusso di informazioni omogenee in movimento da un processo ad un altro.

Il processo è definito come un insieme di operazioni che trasformano flussi di dati (data flow) in ingresso in flussi di dati in uscita.

Il data store rappresenta un insieme di dati temporaneamente "fermi", in attesa di essere utilizzati. Sono assimilabili ad archivi, in cui i dati sono immessi da appositi processi, ed utilizzati da altri processi in tempi successivi a quelli in cui sono stati prodotti.

L'Agente esterno (External Agent) è definito come una entità esterna all'area problema che si analizza, con le quali il sistema entra in contatto. Esempi di Agenti esterni possono essere altre unità organizzative, oppure gruppi interni alla propria organizzazione, ma esterni all'area problema in esame.

Nel DFD non vanno evidenziate eventuali relazioni tra le Entità Esterne.

I DFD possono essere sviluppati gerarchicamente come serie di livelli di informazioni.

Il primo passo consiste nell'effettuare le seguenti attività:

- Analizzare il testo
- Identificare i dati
- Identificare le sorgenti
- Identificare le destinazioni
- Costruire una tabella riassuntiva del seguente tipo:

Sorgente	Dati	Destinazione

Il secondo passo consiste nel costruire il diagramma di contesto, noto anche come Diagramma di livello 0 in cui è riportato un solo processo e le entità esterne ad esso associate.

Il terzo passo consiste nell'individuazione dei confini del sistema; e ciò avviene mediante l'esame delle entità riportate nella precedente tabella ed in base alla decisione sulla loro tipologia di interne od esterne.

Una volta identificati i confini:

- Tutto ciò che è all'interno dei confini è un processo
- Vengono fissate le entità esterne ed interne
- Le relazioni del sistema con l'ambiente vengono indicate dal data flow da/per le entità esterne.

A volte bisogna scendere di livello per capire meglio le cose e poi tornare indietro a correggere i diagrammi più generali, per cui necessità giungere al Diagramma di livello 1.

I diagrammi di livello 1 modellano processi e data stores generali.

Le regole sono:

- Preservare le relazioni sistema – esterno
- Identificare solo processi e archivi generali
- Identificare un numero minimo di processi per gestire i flussi in/out
- Consultare la tabella per identificare data flow e data stores mancanti.

Spesso necessità approfondire ulteriormente il livello per cui si produce il Diagramma di livello 2 in cui ogni processo di livello 1 ed i suoi input ed output possono essere considerati il diagramma di contesto per ogni componente del sistema e si riapplicano le stesse tecniche

E così via livello dopo livello.....

5. IL ERD (ENTITY RELATIONSHIP DIAGRAM)

Il modello concettuale dei dati ormai affermato come standard di riferimento, è il modello Entity – Relationship, il cui schema è il ERD.

La modellazione dei dati deve permettere di stabilire quali sono i principali oggetti che il sistema deve manipolare, quali sono gli attributi che descrivono questi oggetti, come questi oggetti sono composti, come sono in relazione tra loro e tra gli oggetti e i processi che li trasformano. Queste informazioni vengono rappresentate nell'ambito della progettazione strutturata tramite l'ERD – Entity-Relationship Diagram.

Il modello entità/associazioni (E/R = Entity/Relationship), introdotto nel 1976 da Peter P. Chen, è uno strumento per analizzare le caratteristiche di una realtà in modo indipendente dagli eventi che in essa accadono, cioè per costruire un modello concettuale dei dati indipendente dalle applicazioni.

Gli elementi di un modello E/R sono:

- Entità che rappresenta classi di oggetti del mondo reale (persone, cose, eventi, ...), che hanno proprietà comuni ed esistenza autonoma;
- Associazioni (Relationship) che è un legame che stabilisce un'interazione tra le entità;
- Attributi che caratterizzano le proprietà dell'entità;
- Generalizzazioni che descrivono un collegamento logico tra un'entità detta entità padre e una o più entità dette entità figlie e

Sottoinsiemi che è un caso particolare di generalizzazione con una sola entità figlia.

Solitamente i *sostantivi* che compaiono nelle frasi del linguaggio naturale corrispondono alle entità, mentre i *verbi* corrispondono alle associazioni, le proprietà delle entità e delle associazioni descrivono gli attributi. Ad es. Una persona (Entità) possiede (associazione) una macchina (Entità) di 100.000Euro (Attributo).

6. IL PROTOTIPO (PROTOTYPING)

La tecnica del *prototyping* si sta affermando grazie al fatto che permette di mostrare in modo iterativo all'utente ciò che il progetto vuole realizzare.

In effetti, le basi culturali e la visione del mondo del committente e degli utenti di software gestionale, molto diverse da quelle dei tecnici, spesso rendono difficile la comunicazione e la comprensione, tanto da consigliare l'utilizzazione di tutti gli strumenti disponibili per diminuire le ambiguità che poi portano alla realizzazione di prodotti diversi da quelli pensati, e quindi attesi, dal committente.

Il prototipo di analisi stabilisce un protocollo di comunicazione comune tra l'utente e l'analista. Questo permette all'utente di esprimere interamente le proprie esigenze e all'analista di approfondire la conoscenza del dominio del problema.

Può quindi essere utile sperimentare approcci meno formali e verificare la loro potenza in termini di risultati e di economie.

Il prototipo non deve mai evolversi nel prodotto finito e a tale scopo è preferibile che sia realizzato con strumenti diversi da quelli che si utilizzeranno in fase di realizzazione. La sua funzione è solo quella di mostrare agli utenti ai vari livelli il comportamento del sistema, al fine di ottenere da essi il maggior numero di informazioni possibili per ampliare la conoscenza del dominio del problema.

Nelle fasi di analisi, il prototipo ha l'obiettivo di portare la conoscenza della realtà d'interesse a livello di dettaglio completo; in particolare serve ad individuare contemporaneamente le caratteristiche statiche (attributi delle entità) e dinamiche (flusso logico funzionale) di ciascuna funzionalità individuata nell'analisi e definizione dei requisiti. Il prototipo quindi rappresenta un modello empirico utilizzato per la realizzazione e la formalizzazione dei modelli concettuali, ed è lo strumento che aiuta a raffinare i diagrammi E-R e DFD.

Il prototipo è creato nel rispetto delle funzionalità di primo livello individuate, a partire dai requisiti formalizzati. La realizzazione del prototipo avviene attraverso successivi raffinamenti, fino ad ottenere uno schema concettuale esauriente, sul quale elaborare le modalità di realizzazione del sistema durante la fase di progettazione. Alla fine

dell'attività di analisi deve essere disponibile un prototipo di livello n, derivante da n raffinamenti, dipendentemente dalla complessità del dominio del problema, che descrive completamente l'interfaccia utente. L'analista ha così la possibilità di presentare all'utente il prodotto, nelle sue prime fasi di realizzazione, e l'utente di valutare se tale prodotto corrisponde alle sue esigenze.

7. IL DOCUMENTO D'ANALISI

Il Documento di analisi ha la seguente organizzazione:

Premessa: Descrizione della realtà d'interesse; Contesto; Vincoli; Ambiente Tecnologico; Sistema Operativo; DBMS; Ambiente di Sviluppo; Interfaccia.

Obiettivi: Descrizione degli obiettivi che il software da sviluppare deve perseguire.

Funzionalità da implementare.

Il compito del documento è quello di specificare il dominio del problema al livello di dettaglio completo, senza preoccuparsi della definizione del dominio, che è già stata affrontata nelle fasi precedenti

. Per questo motivo, nel documento di analisi non esiste una sezione dedicata alle funzionalità esistenti e quelle future, ma verranno trattate solo le funzionalità da implementare.

Siccome in questa fase viene approfondita la conoscenza del dominio del problema, nel documento di analisi vengono dettagliate le descrizioni delle funzionalità individuate in fase di analisi e definizione dei requisiti.

Infine nel documento di analisi verrà creato un Dizionario dei Dati, come documentazione dello schema concettuale dei dati. Tale dizionario è composto da due tabelle, una per la descrizione delle entità, l'altra per quella delle relazioni.

III METODI E TECNICHE PER L'ANALISI TECNICA

1. INTRODUZIONE

L'analisi tecnica permette di definire e descrivere compiutamente la struttura fisica del prodotto software da realizzare, utilizzando metodi e tecniche di scomposizione e modellizzazione atti ad organizzare le informazioni per la successiva attività di codifica.

In pratica nella progettazione concettuale si era visto "che cosa" rappresentare, ora si tratta di definire "come" realizzare quanto individuato, nella fase precedente.

Due sono gli aspetti del sistema che vanno descritti in questa fase:

- statici (progettazione logica dei dati);
- dinamici (progettazione logica delle funzionalità).

L'analista riceve dal team di lavoro che ha analizzato le funzionalità da soddisfare, una serie di prodotti indispensabili che sono il punto di partenza per la sua attività: il documento di analisi, lo schema Entità - Relazione e il Data Flow Diagram delle funzioni.

Dallo schema concettuale dei dati (ERD) l'analista elabora la struttura della base dati, con il supporto del modello relazionale. In questo momento deve essere in grado di valutare e prevedere il carico applicativo e tenere conto dell'ambiente di sviluppo scelto in fase di analisi e definizione dei requisiti. Egli realizza un diagramma logico dei dati che è radicalmente diverso dallo schema ERD di partenza, infatti vengono fatte considerazioni sulle dimensioni dei campi, sul numero occorrenze e sulle distribuzioni delle tabelle nei diversi *Database*.

Dallo schema concettuale delle funzioni (DFD) l'analista espone in maniera dettagliata tutte le operazioni che creano una corrispondenza tra la realtà d'interesse ed il sistema informatico ad un punto tale da permettergli di prendere in considerazione la successione temporale e la sincronizzazione degli eventi attraverso lo schema di struttura Stati/Eventi.

Il raffinamento del documento di analisi e la sua rielaborazione a livello di singola funzionalità, descritta mediante i DFD e diagrammi Stati/Eventi, unitamente ad un efficace utilizzo del linguaggio naturale, producono il documento delle Specifiche di Progetto.

Infine la tecnica del prototyping permette di unire in modo intuitivo la funzionalità da implementare con la base dati secondo una vista logica.

2. IL MODELLO RELAZIONALE

Nella progettazione della parte statica del sistema l'obiettivo è quello di costruire uno schema logico in grado di descrivere, in maniera corretta ed efficiente tutte le informazioni contenute nel diagramma E-R prodotto nella fase concettuale.

Lo strumento che si utilizza in questa fase è il modello relazionale.

Non si tratta di una semplice traduzione da un modello a un altro perché, prima di passare allo schema logico, lo schema E-R va "ristrutturato" per soddisfare le esigenze di tradurre e di ottimizzare le prestazioni del progetto.

La progettazione logica dei dati deve tenere conto, per quanto possibile, delle prestazioni dell'applicazione adattandosi al sistema di basi di dati (DBMS) scelto.

I dati in ingresso della prima fase sono lo schema E-R prodotto nella fase concettuale e il carico applicativo previsto, in termini di volume dei dati e frequenza delle operazioni.

Il risultato che si ottiene è uno schema E-R ristrutturato, che non è più uno schema concettuale tout court, in quanto costituisce una rappresentazione dei dati che tiene conto degli aspetti realizzativi.

La prima cosa da fare è un'analisi delle prestazioni, per l'ottimizzazione relativa gli indici di prestazione che sono:

- costo di una operazione;
- occupazione di memoria.

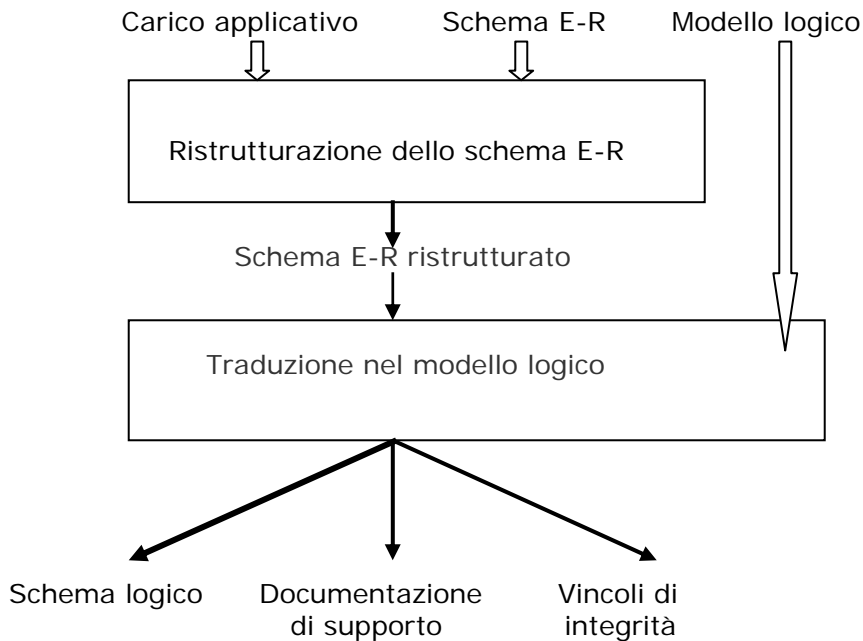
All'analisi delle prestazioni segue la ristrutturazione del modello E-R. Le fasi più importanti sono:

- l'analisi delle ridondanze (informazioni significative ma derivabili da altre e quindi inutili, salvo alcuni casi particolari di ridondanze obbligate) in cui si decide se eliminare le ridondanze presenti o mantenerle, ;
- l'eliminazione delle generalizzazioni mediante la loro sostituzione con entità e relazioni direttamente rappresentabili nel modello relazionale;
- partizionamento/accorpamento di entità e relazioni effettuate per rendere più efficienti le operazioni di accesso in base ad un semplice principio di una loro riduzione;
- la scelta degli identificatori principali per la traduzione nel modello relazionale nelle chiavi primarie.

A questo punto si è in grado di disegnare lo schema E-R ristrutturato tenendo presente tutte le considerazioni suesposte.

Disegnato lo schema E-R ristrutturato si passa alla seconda fase in cui i dati in ingresso sono lo schema E-R ristrutturato ed il modello logico prescelto da cui si costruisce uno schema logico equivalente che rappresenti le medesime informazioni.

Nella figura seguente si illustrano le varie fasi della progettazione per la traduzione dallo schema concettuale a quello logico e si evidenziano gli strumenti e le informazioni di input di cui si tiene conto.



Lo schema logico finale, i vincoli di integrità definiti su di esso, cioè le proprietà che devono essere soddisfatte da tutte le istanze corrette della base dati, e la relativa documentazione costituiscono i prodotti finali della progettazione logica

3. DFD e Stati/Eventi

Nella progettazione della parte dinamica del sistema si parte dalla trasformazione delle informazioni delle funzionalità acquisite dall'analisi. Da un lato il progetto risultante deve essere sufficientemente astratto per essere agevolmente confrontato con le specifiche da cui viene derivato, dall'altro lo stesso progetto deve essere sufficientemente dettagliato in modo tale che la codifica possa avvenire senza ulteriori necessità di chiarire le operazioni che devono essere realizzate.

Per muoversi dall'organizzazione a grafo dell'analisi (DFD) ad un'organizzazione tecnica per il progetto si possono utilizzare nuovamente come strumento i DFD scendendo ad un livello di dettaglio in cui viene descritto ciascun processo semplice che appartiene ad una funzionalità.

Mentre i DFD dell'Analisi sono una rappresentazione reticolare di bolle e memorie di massa, i diagrammi strutturati per la progettazione sono una rappresentazione gerarchica di moduli in cui si sono aggiunte considerazioni basate su vincoli realizzativi. E' sottinteso che in ogni passo di raffinamento deve essere assolutamente rispettato il vincolo di continuità del flusso informativo, poiché per progetti particolarmente complessi il diagramma rischierebbe di diventare poco controllabile.

I DFD forniscono solamente informazioni sulle dipendenze funzionali tra i dati, ma sono quantomeno imprecisi per quanto riguarda gli aspetti di sincronizzazione e controllo, pertanto dei processi elementari che costituiscono la funzionalità di progetto, descritta nei DFD, si rappresenta lo stato per mezzo di diagrammi Stati/Eventi: questa tecnica introduce, per ciascuna funzionalità, la coordinata temporale nella descrizione dei vari processi, che risulteranno così sincronizzati.

Lo stato di ogni funzionalità dipende dalle sequenze di operazioni che si verificano in seguito a stimoli (eventi) esterni ed il diagramma Stati/Eventi descrive come una funzionalità risponde ad un determinato evento quando si trova in un particolare stato.

Tutti i possibili stati in cui una funzionalità può trovarsi e tutti i possibili eventi che possono generare una reazione di una funzionalità sono riassunti nel diagramma degli Stati/Eventi, che corrisponde ad un grafo in cui i nodi sono gli stati e gli archi sono gli eventi.

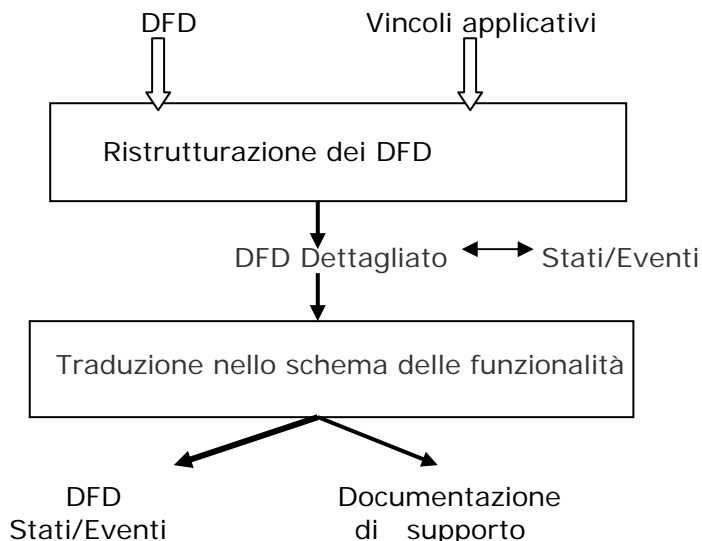
Riassumendo, la progettazione delle funzioni a partire dal DFD di livello 1 viene esplosa dettagliatamente per ogni funzionalità.

Inoltre per ciascuna funzionalità viene sviluppato il diagramma stati/eventi e, se necessario, è possibile rappresentare alcune funzioni utilizzando i diagrammi di flusso o la pseudocodifica.

Nella figura seguente si sintetizzano le fasi della progettazione logica delle funzionalità.

Il DFD dettagliato con relativo schema Stati/Eventi, e la relativa documentazione costituiscono i prodotti finali della progettazione logica delle funzionalità.

Il raffinamento del documento di analisi e la sua rielaborazione a livello di singola funzionalità, descritta mediante i DFD e diagrammi Stati/Eventi producono il Documento delle Specifiche di Progetto integrato sino al sottosistema di primo livello.



4. Il prototipo (Prototyping)

Allo scopo di fornire uno strumento al progettista che si basa su una descrizione formale meno astratta e più efficientemente eseguibile dei requisiti dettati dall'analisi si introduce la tecnica della prototipizzazione (prototyping) per determinare la reale fattibilità delle funzionalità.

In questa ottica si definisce un modello di prototipo il cui solo obiettivo è quello di contribuire a chiarire allo sviluppatore come devono essere realizzate le specifiche definite nei documenti di Analisi.

Nel prototipo di progetto sarà trascurato l'effettivo svolgimento delle operazioni che sono comandate attraverso le maschere, ma per ciascun pulsante o casella o strumento tipico degli ambienti visuali sarà specificato mediante linguaggio naturale l'aspetto tecnico vero e proprio che supporta il processo.

Lo sviluppatore che si trova a dover considerare una per una le funzionalità per svilupparle, trae tutte le informazioni dal documento di progetto ed in particolare dagli schemi DFD e Stati/Eventi, inoltre avendo a disposizione il prototipo di progetto ha una visione rapida e completa di come dovrà apparire il prodotto finale.

5. Il documento di progetto

Gli strumenti introdotti conducono alla stesura, a mezzo del linguaggio naturale, del documento forse più importante di tutta l'attività: il documento di progettazione in cui devono essere dichiarate in modo esplicito le scelte progettuali effettuate.

Il contenuto del documento deve comprendere:

- **Premessa:** Descrizione dell'ambito in cui il lavoro si colloca; Contesto; Vincoli; Ambiente tecnologico; Sistema Operativo; DBMS; Ambiente di Sviluppo; Interfaccia.
- **Obiettivi:** Descrizione degli obiettivi che il prodotto si prefigge.
- **Lavoro di Progetto.**

Progettazione della Base Dati: In questa sezione il progettista inserisce il diagramma logico necessario alla realizzazione della base dati con commento in linguaggio naturale.

Egli illustra le scelte progettuali che lo hanno portato ad organizzare una base dati singola o distribuita.

Progettazione Funzionale: In questo paragrafo si inseriscono gli schemi DFD per ogni singola funzionalità al livello di astrazione più basso. Va indicato un sottoparagrafo per ciascuna esplosione del DFD, così da poter essere individuato direttamente anche dall'indice. Ove necessario (flussi funzionali particolarmente complessi o temporalmente collegati) va riportato lo schema Stati/Eventi.

Realizzazione delle Funzionalità: Gli ambienti di sviluppo visuale consentono una rapida visualizzazione delle funzionalità a mezzo del

prototipo. Questa sezione del documento è dedicata ai commenti in linguaggio naturale concernenti la realizzazione delle funzionalità.

Dizionario dei Dati: fornisce una descrizione squisitamente sintattica (nome, tipo, lunghezza, descrizione) dei campi, dal momento che specifica esclusivamente la struttura, tralasciando il significato (peculiare dell'analisi).

IV METODI E TECNICHE PER LO SVILUPPO

1. INTRODUZIONE

Lo sviluppo di un sistema informatico è la fase conclusiva della produzione del software gestionale.

Il presente documento si propone di tracciare una panoramica sulle tecniche di Sviluppo di software di tipo gestionale.

2. TECNICHE DI PROGRAMMAZIONE

L'utilizzo dei linguaggi visuali porta naturalmente ad acquisire le tecniche di programmazione modulare, finalizzate alla riduzione della complessità dei problemi da risolvere oltre che al riutilizzo ed alla manutenzione dei programmi, grazie all'associazione modulo-funzionalità.

2.1 Programmazione Modulare

Il metodo di programmazione modulare ha come obiettivo la riduzione della difficoltà nel comprendere e risolvere i problemi.

L'idea che sta alla base della programmazione modulare è quella di ridurre la complessità di un problema suddividendolo in sottoproblemi. La programmazione modulare parte dal presupposto che, suddividendo correttamente un programma in moduli, anche considerando la complessità delle loro interrelazioni, la complessità globale risulta ridotta.

La modularizzazione di un programma solleva due questioni; la prima di carattere quantitativo (sulle dimensioni dei moduli) e la seconda di carattere qualitativo (sui criteri di suddivisione dei moduli).

Per quanto riguarda gli aspetti quantitativi al diminuire delle dimensioni diminuisce la complessità di un modulo, ma aumenta il numero dei moduli e quindi la complessità delle loro interrelazioni.

Sulla qualità del processo di suddivisione di un programma in moduli, si tratta di introdurre dei criteri che rendano minima la complessità dei moduli e delle loro interrelazioni.

Un modulo, identificato da un nome, consiste in un insieme di istruzioni, con un insieme di variabili locali la cui esecuzione può essere richiamata attraverso tale nome. Nel modulo possiamo distinguere:

- la funzione, cioè cosa fa il modulo;
- la logica, in altre parole come il modulo svolge la sua funzione;
- il contesto, cioè l'uso che viene fatto della funzione.

L'importanza della distinzione di questi tre concetti consiste nel fatto che quando si descrive un singolo modulo si deve far sempre riferimento alla sua funzione e non alla sua logica o al suo contesto.

E' importante inoltre conoscere la coesione di un modulo ossia il grado di interdipendenza tra fra le sue singole parti (op. elementari o

istruzioni) componenti e l'accoppiamento ossia la misura delle interrelazioni che esistono fra i moduli di un programma.

L'obiettivo della metodologia di programmazione modulare è quello di ottenere il minimo accoppiamento e la massima coesione.

3. GLI ALGORITMI

Che cos'è un algoritmo

L'elaboratore è una macchina elettronica in grado di svolgere in modo estremamente veloce un gran numero di operazioni; ma quello che non può fare è agire di sua spontanea volontà; in effetti essendo una macchina possiamo considerarlo nient'altro che un semplice esecutore privo della benché minima capacità decisionale; lo si può perciò paragonare a una persona il cui compito è esclusivamente quello di obbedire agli ordini senza prendere nessuna iniziativa.

Un **algoritmo** è la descrizione della sequenza di operazioni che portano alla soluzione di un problema.

Nella descrizione di un algoritmo bisogna rispettare alcune regole:

- Condizione di eseguibilità secondo cui l'esecutore deve essere in grado di eseguire l'istruzione.
- Condizione di finitezza secondo cui l'algoritmo deve poter essere eseguito in un tempo finito;
- Condizione di non ambiguità secondo cui l'esecutore è per definizione privo di iniziativa quindi fornire un comando vago o che si presti a diverse interpretazioni non è consentito.

Algoritmi a prima vista corretti possano portare a risultati non desiderati, ciò dovrebbe farci capire l'importanza di studiare attentamente il problema e valutare l'effetto reale delle istruzioni che compongono l'algoritmo.

Non esistono strumenti automatici che siano in grado di verificare che un algoritmo raggiunga il risultato voluto e quindi bisogna eseguire questo compito manualmente (in altre parole non esistono algoritmi che verificano il funzionamento di algoritmi).

Nella descrizione degli algoritmi si parla di **livello di dettaglio** facendo riferimento al fatto che per semplicità talvolta essi non spiegano per filo e per segno quali sono esattamente le operazioni da compiere, ma piuttosto forniscono una linea d'azione più generale.

E' però da evidenziare che gli algoritmi sono semplici descrizioni che poi andranno tradotte in un linguaggio idoneo all'esecutore e che, per questo, è necessario passare da un livello di scarso dettaglio (quello dell'algoritmo) ad un livello più particolareggiato (quello del linguaggio). A questo scopo si usa procedere per raffinamenti successivi, fino a giungere al massimo livello di dettaglio necessario.

Dal punto di vista dell'esecutore l'algoritmo è più correttamente indicato col nome di programma; per passare da un algoritmo a un programma bisogna fare uso di un linguaggio di programmazione.

Il programma non è altro che la sequenza di istruzioni da eseguire espresse nel linguaggio dell'esecutore (per gli elaboratori si parla di linguaggio macchina).

Rappresentazione degli algoritmi

Gli algoritmi, e quindi i programmi, sono fatti di sequenze di istruzioni da eseguire secondo un percorso che può procedere in maniera lineare, ciclica o ramificata. In effetti le tre strutture di controllo di flusso che si ritrovano in tutti i linguaggi di programmazione (o quasi), sono la struttura sequenziale, quella iterativa (ciclo) e quella condizionale.

Affinchè tali elementi siano evidenziati nella descrizione degli algoritmi, è necessario usare dei metodi di rappresentazione, degli algoritmi stessi. I metodi più utilizzati per questo scopo sono tre: il flow chart, la pseudocodifica, il linguaggio di programmazione.

3.1 Il Flow Chart (Diagramma a Blocchi)

I Flow Chart sono dei diagrammi che, per mezzo di alcuni simboli convenzionali, consentono la descrizione grafica degli algoritmi.







Ogni algoritmo di una certa complessità deve essere documentato con un Flow Chart, eventualmente esploso su più livelli di astrazione affinché la modularizzazione funzionale possa già affiorare in questa fase alta dell'implementazione.

Di convenzioni sul significato dei vari simboli non ne esiste solo una ma, anzi, talvolta lo stesso simbolo può essere usato con significati diversi da persone che applicano convenzioni differenti, oppure alcuni utilizzano dei simboli diversi per specificare la stessa cosa. Per i nostri scopi, comunque, sarà necessario un ristretto insieme di tali simboli il cui significato è, più o meno, universalmente accettato.

Tali simboli sono connessi tra di loro per mezzo di frecce che indicano, pertanto, la direzione del flusso. Talvolta, in letteratura, il punto d'inizio e quello di fine di un algoritmo sono segnalati tramite due simboli speciali Start ed End:



Quella che segue è tabella dei simboli grafici di cui faremo uso d'ora in poi.

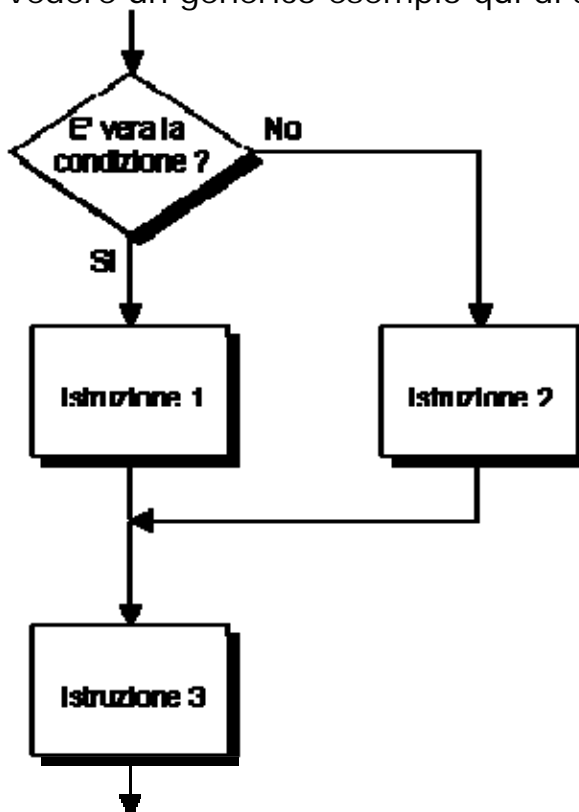
	Sequenza	Indica l'istruzione da eseguire. Ha solo una freccia in uscita.
	Condizione	E' normalmente utilizzato per valutare una condizione che può valere vero o falso (SI o NO). Due frecce in uscita, una per il valore e l'altra per il falso. E' alla base delle strutture condizionali e iterative.
	Rimando a pagina successiva	Usato quando un flow chart non entra in un'unica pagina come segnale di continuazione. Potendo essercene più d'uno vengono identificati da simboli (di solito lettere). Non ha uscite.
	Output	Rappresenta un output generico. Ha solo una freccia in uscita. Il messaggio può essere anche una variabile.
	Input	Rappresenta un input generico. Ha solo una freccia in uscita.
	Chiamata a sub routine	Usato per indicare la chiamata a sub routine con passaggio di parametri. Il valore ritornato può essere memorizzato in una variabile. Ha solo una freccia in uscita.

Vediamo degli esempi delle strutture di controllo di flusso rappresentate tramite flow chart.

Iniziamo con la più semplice, ossia con la sequenza che non è altro che una successione di istruzioni che vanno eseguite una dopo l'altra, un esempio di tale struttura è mostrato nella figura seguente:



Le strutture condizionali, invece, sono necessarie quando ci si presenta un'alternativa cioè nelle occasioni in cui se si verifica una certa condizione bisogna agire in un modo, altrimenti in un altro. Ne possiamo vedere un generico esempio qui di seguito:

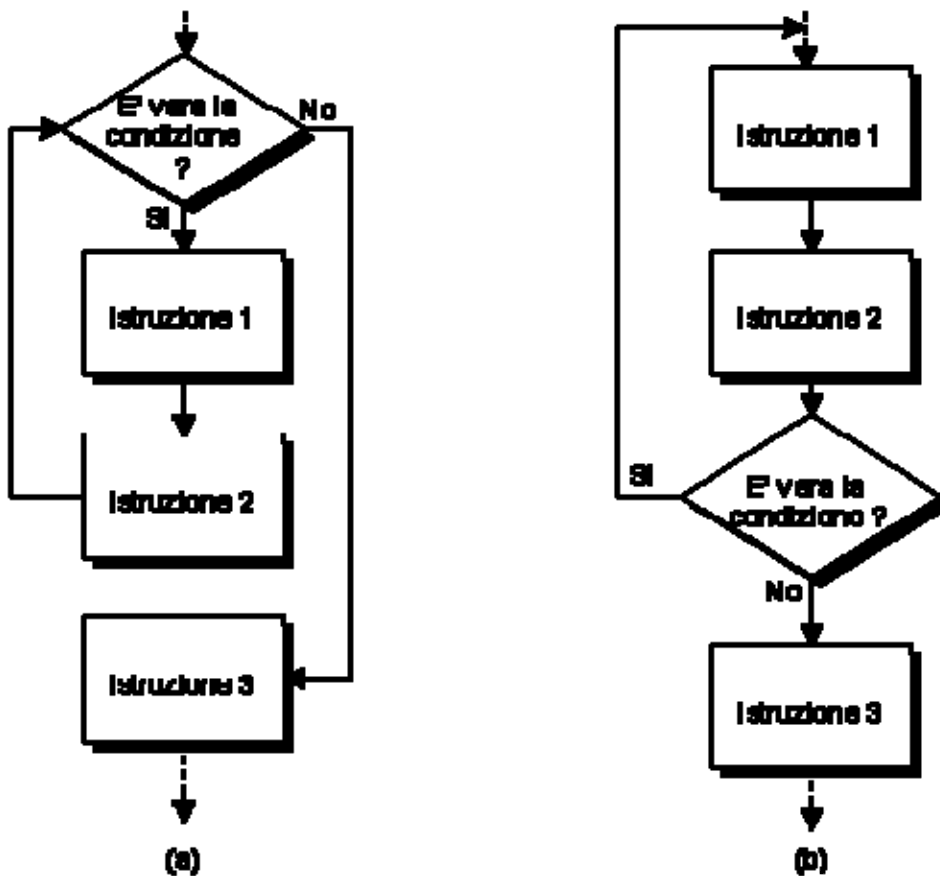


E' da notare come, dopo la valutazione della condizione e l'esecuzione della istruzione, o delle istruzioni dal momento che possono essere più

d'una, il flusso del programma prosegue comunque con l'istruzione 3, indipendentemente dal valore della condizione.

Infine la struttura iterativa che consente di eseguire ripetutamente una serie di istruzioni fintantoché una determinata condizione sia vera (o falsa a seconda delle scelte).

La Figura successiva mostra due tipi di iterazione, il primo con la valutazione della condizione all'inizio, il secondo con la valutazione alla fine; la differenza sta unicamente nel fatto che se nel primo tipo la condizione è subito falsa non viene eseguita neppure una iterazione ma si prosegue immediatamente con la prima istruzione fuori del ciclo; nel secondo tipo di iterazione, invece, è garantita almeno una iterazione visto che la condizione è valutata in fondo.



(a) Es. di valutazione immediata della condizione

(b) Es. di valutazione posticipata

Nel caso di algoritmi di una certa complessità si riempiono pagine e pagine di simboli ed in casi come questo, pur usando dei simboli di connessione fra le varie pagine, si perde il colpo d'occhio sul flusso dell'algoritmo, neutralizzando uno dei vantaggi principali dei flow chart per molti preferiscono usare un altro tipo di rappresentazione chiamata pseudocodifica.

3.2 La Pseudocodifica

La pseudocodifica si pone ad un livello intermedio di astrazione: non usa ancora le parole del linguaggio di programmazione ma ne utilizza la sintassi.

E' un passo spesso trascurato nello sviluppo del software, ma i suoi benefici sono evidenti se si pensa alla distribuzione di compiti che consente: uno sviluppatore esperto che non conosce un particolare linguaggio di programmazione può comunque scrivere un programma in pseudocodifica e passarlo a chi ha le competenze necessarie per la traduzione.

In questo contesto, diversi livelli di astrazione possono essere rappresentati raggruppando in routine gruppi di istruzioni, secondo le indicazioni della programmazione modulare.

La pseudocodifica consiste nella descrizione dell'algoritmo in maniera testuale, senza perciò fare uso di simboli grafici, che metta in evidenza le strutture di controllo del programma.

La descrizione testuale dell'algoritmo viene poi tradotta con pochissimo sforzo in molti dei linguaggi più usati, come Pascal, Clipper e C tanto per citare i più usati.

Rimane, comunque, il fatto che il metodo grafico del flow chart può essere più intuitivo e comprensibile per un principiante di quanto non sia la pseudocodifica.

3.3 Il linguaggio di programmazione

Affinché un dato problema possa venire risolto da un elaboratore è necessario che "qualcuno" definisca il relativo algoritmo risolutivo; tale algoritmo è scritto in linguaggio naturale, anche se codificato con i formalismi precedentemente trattati.

Gli algoritmi, per poter essere compresi e quindi eseguiti dai circuiti del computer, devono essere scritti in linguaggio macchina; tale linguaggio è molto lontano dal modo di pensare dell'uomo e molto vicino alla struttura fisica del computer.

I dati su cui operare e le istruzioni da eseguire sono stringhe binarie (sequenze di 0 e 1), si capisce quindi come la "traduzione" degli algoritmi in tale linguaggio sia prerogativa di pochi esperti che conoscono a fondo i circuiti della macchina che deve eseguire le operazioni.

Proprio per poter dare alla macchina, con una serie di comandi "vicini" al linguaggio naturale, degli ordini ad essa "comprensibili" sono nati i **linguaggi di programmazione**; questi sono linguaggi formali costituiti da parole in genere non difficili da ricordare (grammatica del linguaggio), combinate secondo rigide regole grammaticali (sintassi del linguaggio).

Per utilizzare un linguaggio di programmazione bisogna conoscere quindi la sua sintassi, la sua grammatica e la sua semantica.

Da quanto detto risulta chiaro che un **programma** è la **traduzione** di un algoritmo in un **linguaggio di programmazione**.

Negli ultimi cinquant'anni sono stati fatti passi da gigante, oggi esistono infatti dei linguaggi molto vicini alla sintassi del linguaggio naturale, ma è chiaro che quanto più questi linguaggi risultano comprensibili, tanto più essi si allontanano dal linguaggio macchina.

Per interfacciare i linguaggi simbolici con il linguaggio macchina si sono sviluppati nel tempo dei programmi che traducono il programma simbolico (programma sorgente) in linguaggio macchina (programma oggetto).

I linguaggi evoluti o ad alto livello e sono frutto di un progressivo tentativo di avvicinamento del linguaggio macchina al linguaggio naturale, sono nati verso la fine degli anni '50 e sfruttano le tecniche della programmazione strutturata la quale nella costruzione degli algoritmi, e quindi dei programmi, fa uso sistematico della sequenza della selezione e dell' iterazione.

I principali programmi procedurali evoluti, tutt'oggi in uso, sono:

- FORTRAN (FORmula TRANslator): nato nel 1956 ed usato per scopi scientifici ed ingegneristici è stato il primo linguaggio ad alto livello
 - COBOL (COMmon Businnes Oriented Language): nato nel 1960, ancora in uso per la sua semplicità ed elasticità, per applicazioni commerciali e gestionali
 - Pascal: nato nel 1971 in ambiente prettamente scientifico, molto usato anche dal punto di vista didattico per l'uso coerente dei principi della programmazione strutturata
 - C: nato nel 1974 sempre in ambito scientifico, con struttura aderente ai principi della programmazione strutturata, è molto diffuso e permette di scrivere programmi per risolvere problemi molto complessi.
- Altri linguaggi molto usati furono il LISP, il Prolog, e l'ADA, anche questi sviluppati dal '60 alla fine degli anni '70.

I linguaggi evoluti, nati a partire dagli anni '70, non sono linguaggi strutturati nel senso che il flusso dei dati non può essere rappresentato per mezzo delle codifiche sopraesposte; in questo tipo di programmi il flusso delle informazioni non è "rigido", "prevedibile" ma varia a seconda degli eventi generati dall'azione dell'utente sugli oggetti. Esistono versioni ad oggetti del C e del Pascal, rispettivamente C++ e Turbo Pascal ; esempio di linguaggio ad eventi è il Visual Basic di linguaggio ad oggetti Power Builder.

V METODI E TECNICHE PER LA MISURAZIONE DEL PRODOTTO SOFTWARE

1. INTRODUZIONE

“Quando puoi misurare quello di cui stai parlando, e puoi esprimerlo con dei numeri, lo conosci abbastanza; ma quando non puoi misurarlo, quando non puoi esprimerlo con dei numeri, la tua conoscenza è scarsa ed insoddisfacente.

Lord Kelvin, 1889”

Esistono molte ragioni per misurare la dimensione del software, proprio come ci sono molte ragioni per misurare la superficie di una casa. In un particolare contesto, potrebbe essere necessario misurare la dimensione attesa del software prima del suo sviluppo, proprio come potrebbe essere necessario determinare la superficie di una casa prima della sua costruzione. In un differente contesto, sarà utile misurare la dimensione del software *a posteriori*, cioè un certo tempo dopo che esso è stato messo in produzione, proprio come potrebbe essere utile misurare la superficie di una casa dopo che essa è stata costruita e il proprietario vi si è trasferito.

L'obiettivo del presente documento è quello di fornire nozioni su alcuni metodi e tecniche utilizzate per la misurazione del prodotto software.

2. LE DEFINIZIONI

In qualsiasi processo produttivo è importante poter misurare; infatti la finalità di una misura è di rendere oggettivi i risultati ottenuti a seguito dell'espletamento delle specifiche attività, in modo da poter rendere un determinato processo ripetibile e confrontabile.

Possiamo quindi fornire le seguenti definizioni:

- la **MISURAZIONE**, è un processo che assegna numeri o simboli ad attributi di entità del mondo reale, per descriverle secondo regole ben definite, ed ha come oggetto la determinazione del valore di una misura. Un esempio di entità è la persona, il semaforo, un programma. Un esempio di attributo è l'età, il colore del semaforo, la dimensione di un programma. Un esempio di valori è 18 anni, rosso, 35 linee di codice;

- la **MISURA**, è il risultato di una misurazione ed è una assegnazione empirica ed oggettiva di un numero o simbolo ad una entità per caratterizzarne un attributo specifico.

- la **METRICA**, è un insieme di regole per stabilire le entità da misurare (es. persona), gli attributi rilevanti e le unità di misura (es. altezza), le procedure per assegnare numeri e simboli (es. 180 cm).

Secondo la norma internazionale ISO/IEC 9126, le caratteristiche classificano gli aspetti ritenuti essenziali (e non sono misurabili), gli attributi (o sottocaratteristiche) rappresentano una classificazione più fine delle caratteristiche (non misurabili), e gli indicatori che sono proprietà misurabili degli attributi. Gli indicatori vengono usati per misurare caratteristiche difficilmente misurabili, infatti le metriche per stimare caratteristiche permettono di identificare gli attributi misurabili e di stimare le caratteristiche non misurabili; ad esempio la dimensione è un attributo misurabile, mentre il costo di manutenzione è una caratteristica stimata.

E' da precisare infine che spesso non si distingue fra entità ed attributi, infatti spesso si dice: "Antonio è più alto di Paolo", mentre la dizione esatta è: "l'altezza di Antonio è maggiore dell'altezza di Paolo" in quanto non si misurano le entità ma i loro attributi.

3. TIPOLOGIE DI METRICHE

Ogni qual volta si vuole valutare la bontà di un bene o di un servizio risulta pressoché indispensabile esprimere il giudizio attraverso una metrica .

Una metrica, come detto, è una funzione che permette di associare ad un attributo di un entità (per esempio l'altezza di una persona) un valore numerico o simbolico (per esempio il numero 1.80 che esprime in metri l'altezza della persona).

Non esistono solo metriche numeriche. Quando compiliamo un questionario di valutazione di un bene o di servizio ci troviamo spesso di fronte a domande le cui risposte sono codificate attraverso un insieme di valori predefiniti; per esempio, se ci viene chiesto di valutare la cortesia e efficienza di un servizio di assistenza all'utente, tipicamente le risposte possibili sono "ottimo", "buono", "sufficiente" e "insufficiente"; questa è una metrica che si basa su valori scalari e non numerici.

Nel processo di produzione si individuano quattro tipologie fondamentali di metriche:

- Metriche relative ai **Processi** che riguardano principalmente:
 - le risorse;
 - l'aderenza alla pianificazione;
 - l'impatto sul personale;
 - le statistiche.
- Metriche relative ai **Progetti** che riguardano principalmente le attività concrete legate a tempo e risorse.
- Metriche relative ai **Prodotti** che riguardano principalmente beni e servizi forniti dai progetti.

- Metriche relative alle **Risorse** che riguardano principalmente gli elementi necessari per istanziare progetti.

Inoltre esistono due tipi di attributi per le metriche:

- **Interni**, misurabili in termini delle entità;
- **Esterni**, misurabili in termini di come le entità sono in relazione con l'ambiente esterno.

4. METRICHE PER IL PRODOTTO SOFTWARE

L'uso di metriche per il software è un problema aperto, questo perché un software è difficile da misurare, a causa dei diversi aspetti che vengono presentati, dall'ambiente e dalle tecnologie che permettono di creare un software.

Le metriche per il software sono concepite basandosi sul prodotto in sé (quanto è grande), sulle funzionalità (cosa deve fare) o sul suo comportamento (cosa succede) e sono sostanzialmente di quattro tipi:

1. dimensionali;
2. di complessità;
3. Object Oriented;
4. di qualità.

Le metriche sono sostanzialmente di due tipi:

Diretta, basata su una misura direttamente osservabili(es. linee di codice, tempo totale di sviluppo);

Indiretta: non è direttamente osservabili ma viene effettuata a partire da altre metriche(es. numero di difetti / linee di codice) .

4.1 Metriche dimensionali

Le metriche dimensionali sono quelle basate sul codice; esse valutano le dimensioni di un programma.

Tali metriche sono sostanzialmente di due tipi:

1. Metriche basate sul numero di istruzioni del programma fra le quali la più nota è la **LoC** (Lines of Code – Linee di Codice).
2. Metriche basate sul numero di funzionalità presenti nel programma. L'esempio classico in questo caso è costituito dai **Function Point**.

4.1.1 LoC(Lines of Code - Linee di Codice)

La base della metrica LoC (Line of Code - Linee di Codice) è che la lunghezza del programma può essere usata per prevedere le caratteristiche di un programma come lo sforzo e la facilità di manutenzione. La metrica LoC viene usata per misurare la dimensione

del software; questo tipo di metrica, molto intuitiva, è anche tra le più usate. I metodi più noti sono :

1. DSI;
2. COCOMO

4.1.1.1 DSI

Le DSI sono le "Delivered Source Instructions", basate sulle LoC e definite come segue:

- Soltanto le linee di codice sorgente che verranno incluse nel prodotto finale vengono considerate; programmi di test, debug e di supporto sono escluse;
- Le linee di codice sorgente vengono create dai programmatori; codice creato automaticamente è escluso;
- Una istruzione è una linea di codice;
- Le dichiarazioni vengono considerate istruzioni;
- I commenti vengono considerati istruzioni.

4.1.1.2 Cocomo(CONstructive COst MOdel)

La tecnica COCOMO viene usata per calcolare lo "sforzo" di un progetto software.

COCOMO, è un metodo di stima dello sforzo, di tipo euristico (basato sull'esperienza), sviluppato da B.Boehm (1981).

Stima lo sforzo, in mesi-persona, ed il tempo, in mesi, necessari per sviluppare un prodotto software.

Il metodo permette tre livelli di stima (in base alla quantità di informazioni disponibili):

- Basic (per stime iniziali)
- Intermediate (quando il sistema è stato scomposto in sottosistemi)
- Detailed (quando i sottosistemi sono scomposti in moduli elementari)

4.1.1.3 Cocomo II

Una evoluzione del modello COCOMO è stato il Modello COCOMO II disegnato per:

- Supportare riuso, prototyping e sviluppo di software basato su componenti;
- Considerare la volatilità dei requisiti;

- Considerare differenti granularità nella conoscenza del problema.

4.1.2 – Function Point

Il metodo dei Function Point è stato ideato da Albrecht nel 1979 (presso la IBM), e serve per calcolare le metriche di produttività. Il metodo viene usato in due modi:

1. come una variabile di stima usata per conoscere la "dimensione" di ogni elemento del software;
2. come metrica base collezionata da progetti precedenti ed usate in congiunzione con le variabili di stima per sviluppare proiezioni di costi e "sforzo".

L'approccio è quello di identificare e contare un numero di funzioni tipo uniche:

- Input Esterni (es. filename);
- Output Esterni (es. rapporti, messaggi);
- Interrogazione Esterne (input interattivi che necessitano di risposte);
- File Logici di Interfaccia (file condivisi con altri sistemi software);
- File Logici Interni (invisibili fuori dal sistema).

Ognuna di queste funzioni viene valutata individualmente per la complessità, che può essere Bassa, Media, Alta, e viene assegnato un valore pesato che varia da 3 a 15. Viene poi calcolata la somma di tutte le occorrenze (UFP), moltiplicando ogni conteggio delle funzioni con il valore pesato e, quindi, sommando tutti i valori. I pesi sono basati sulla complessità della funzione contata.

L'individuazione del grado di complessità è effettuata secondo regole specifiche fissate dal metodo, per ogni Funzione Tipo.

Per trovare poi il valore dei punti funzione (FP), UFP viene moltiplicato per il fattore di complessità tecnica, il quale tiene conto di 14 Caratteristiche Generali del Sistema (GSC), valutate sulla base del loro grado di influenza, che varia da quello meno influente al più influente.

4.2 Metriche di Complessità Ciclomatica

Il metodo per calcolare la complessità ciclomatica di un programma è stato inventato da McCabe nel 1976 ed è interamente basato sulla struttura del grafo che rappresenta il programma. La complessità ciclomatica viene definita come la complessità del flusso di controllo, ed è una funzione dei cammini possibili ed indipendenti sul grafo di flusso, inoltre fornisce una rappresentazione astratta del codice ed un valore numerico della complessità.

4.3 Metriche per software Object Oriented

Per il software Object Oriented occorrono delle metriche dedicate, in quanto le metriche tradizionali non sono accurate, per cui sorge il problema tra la complessità funzionale e quella strutturale (che non coincidono), in quanto non c'è linearità nel codice (quindi le LoC non funzionano) e l'uso di strutture e funzioni complesse come McCabe non funziona.

Bisogna quindi studiare delle:

- Metriche per i metodi;
- Metriche per le classi;
- Metriche per i sistemi.

Volendo si può usare McCabe come metrica di base.

4.4 Metriche per la qualità del prodotto.

Le metriche indirette si basano su una misura indiretta la quale presuppone si facciano misure dirette su altri attributi e che queste si combinino per dare una quantificazione di un attributo non direttamente misurabile, in altri termini l'osservazione indiretta si applica quando ci troviamo in presenza di attributi che non possono essere misurati direttamente.

Sebbene non esista una metodologia di classificazione del prodotto software, esistono alcune caratteristiche che lo contraddistinguono; si tratta pertanto di stabilire una metrica che colleghi qualunque interazione misurabile del prodotto con le sue caratteristiche. L'osservazione indiretta del prodotto software, fornisce misure che riflettono indirettamente il suo livello di qualità.

Al tal proposito la norma internazionale ISO 9126 fissa gli attributi di qualità interna ed esterna del software in sei caratteristiche (funzionalità, affidabilità, usabilità, efficienza, manutenibilità e portabilità) a loro volta suddivise in sottocaratteristiche e gli attributi della qualità in uso, categorizzati da quattro caratteristiche (efficacia, produttività, sicurezza, soddisfazione).

Viene fornita la definizione di ogni caratteristica e sottocaratteristica del software che ne influenza gli aspetti di qualità. Per ciascuna caratteristica e sottocaratteristica, la capacità del software viene determinata da un insieme di attributi interni misurabili. Esempi di metriche interne vengono fornite nella norma internazionale ISO/IEC 9126-3. Le caratteristiche e sottocaratteristiche possono essere misurate sulla base della capacità fornita dal sistema che contiene il

software. Esempi di metriche esterne vengono fornite nella norma internazionale ISO/IEC 9126-2.

La qualità in uso è la visione utente della qualità. Il raggiungimento della qualità in uso dipende dal raggiungimento della qualità esterna necessaria, che a sua volta dipende dal raggiungimento della necessaria qualità interna . Sono generalmente richieste misure a tutti e tre i livelli, dato che ottenere i criteri per le misure interne di solito non basta per assicurare il raggiungimento dei criteri per le misure esterne, e raggiungere i criteri per le misure esterne delle sottocaratteristiche di solito non è sufficiente per assicurare il raggiungimento dei criteri per la qualità in uso. Esempi di metriche per la qualità in uso sono dati in ISO/IEC TR 9126-4.

Questo per poter valutare attraverso misura classificazioni e stime il prodotto realizzato.

In genere si definisce un profilo di qualità atteso e poi lo si confronta con un profilo di qualità misurato.

Partendo dai requisiti utente si prevede quella che dovrebbe essere la qualità fornita all'utente e se ne individua un profilo. Una volta sviluppato il prodotto si misura la qualità ottenuta e se ne individua il relativo profilo mediante l'analisi della soddisfazione dell'utente nell'utilizzo del prodotto(**Customer Satisfaction**) e/o il rispetto dei **livelli di servizio** precedentemente concordati. All'uopo vengono definite delle scale ad hoc ed assegnati i valori su di esse (per esempio da 1 a 5 o da scarso ad eccellente) in base alle risposte relative alle specifiche domande riportate in appositi questionari.

Il confronto fra i due profili permette di valutare se il prodotto realizzato ha soddisfatto i requisiti utente.

VI METODI E TECNICHE PER IL TESTING ED IL COLLAUDO

1. INTRODUZIONE

L'attività di testing è parte integrante della fase di sviluppo di un progetto informatico; essa è finalizzata a costituire supporto durante la codifica e l'integrazione delle varie parti del programma in fase di sviluppo e consiste essenzialmente in una serie di controlli pianificati e procedurizzati sul progetto in via di sviluppo, che consentano di espletare correttamente l'attività di controllo interno secondo standard predefiniti.

L'attività di Collaudo è volta a qualificare il prodotto software sviluppato o acquisito nei confronti dei requisiti fissati; il risultato dell'attività di Collaudo è determinante per procedere al rilascio del prodotto software; è quindi necessario definire delle attività di pianificazione, allestimento ed esecuzione delle prove che garantiscano uniformità ed oggettività dei risultati. L'obiettivo del presente documento è quello di fornire indicazioni sui principali metodi e tecniche di Testing e di Collaudo.

2. IL TESTING

Il Testing è una attività mirante alla scoperta di malfunzionamenti del prodotto software dovuti a "guasti" (anomalie – difetti – faults – bugs - bachi) tramite l'esecuzione del codice stesso, fornendogli opportuni dati in ingresso; è da tener presente che il testing può rilevare la presenza di eventuali errori ma non può garantire la loro assenza.

2.1 Principi generali

L'attività di test deve rispettare alcuni principi generali:

- i test vanno condotti tenendo in considerazione i requisiti utente;
- la fase di test va progettata e pianificata in anticipo;
- le attività di test partono dall'analisi delle singole funzioni per coinvolgere l'intera struttura del progetto;
- un'attività di test esaustiva e' impossibile in molti casi (tipicamente esponenziale);
- le attività di test dovrebbero essere condotte da terzi.

2.2 Difetti, errori e malfunzionamenti

Introduciamo alcune definizioni di uso comune nell'attività di test:

- il difetto e' una caratteristica fisica di una porzione di codice o di una sezione di un testo di documentazione;
- l'errore e' l'azione umana che ha generato il difetto;
- il malfunzionamento e' la conseguenza di un difetto che si manifesta durante l'utilizzo del prodotto software.

2.3 Classificazione dei malfunzionamenti

I malfunzionamenti si possono suddividere in quattro categorie:

- Funzionali;
- Strutturali;
- Relativi alla interfaccia;
- Mancato rispetto dei requisiti non funzionali.

2.4 Articolazione del processo di test

Il processo di test si articola secondo i seguenti passi:

- Progettazione dei casi di test;
- Progettazione delle condizioni di test in cui si individuano, tra tutte le possibili situazioni che si possono presentare durante l'utilizzo del prodotto, le sole che si intende riprodurre nel test;
- Individuazione dei casi di test: individuare, per ogni condizione di test e per ciascun dato di input, l'insieme dei valori di tale dato (classe di equivalenza) che rendono vera la condizione;
- Scelta dei valori di input: scegliere per ciascun caso di test (o classe di casi di test equivalenti) uno specifico valore per ciascun dato di input;
- Esecuzione del software per ciascun caso di test e registrazione del comportamento del prodotto;
- Confronto tra il comportamento atteso e quello reale;
- Schedulazione per cui si definisce quando effettuare il testing e quanto sforzo spendere per ogni attività;
- Allocazione personale con l'individuazione del personale responsabile per ogni attività.

2.5 Le Tipologie di Tecniche

Due sono le tipologie di tecniche che vengono considerate per il testing:

- Tecniche statiche, che si occupano di analizzare e controllare le varie rappresentazioni del sistema, ad es., il documento di specifica dei requisiti, o il codice sorgente prodotto;
- Tecniche dinamiche (o test), verificano la bontà della realizzazione

Le Tecniche statiche sono applicabili in tutto il ciclo di vita ed includono analisi del programma e verifiche formali mentre le Tecniche dinamiche, più utilizzate nella pratica, eseguono il programma o il prototipo utilizzando dati simili a quelli che si utilizzeranno una volta installato il sistema.

Se le tecniche disponibili sono basate sull'esame del codice sorgente, allora si parla di Analisi Statica, mentre se ci si basa basate sull'esecuzione del programma allora si tratta di Analisi Dinamica.

2.5.1 Tecniche di Analisi Statica

Le più note tecniche di Analisi statica che temporalmente precedono l'esecuzione dei veri e propri casi di test sono:

- Desk-Checking secondo cui l'esame viene effettuato da una sola persona ricercando la presenza dei difetti contenuti in un' apposita lista e simulando le esecuzioni a cui darebbero luogo specifici valori di input;

- Walkthrough secondo cui il prodotto oggetto di controllo viene esaminato da un gruppo di persone le quali hanno visionato il prodotto prima di incontrarsi in gruppo, vengono individuati alcuni insiemi significativi di valori di input al prodotto ed a fronte di ciascun insieme viene simulata l' esecuzione del prodotto. Il Walkthrough può essere visto come un particolare tipo di seduta di revisione.

2.5.2 Tecniche di Analisi Dinamica

Lo svolgimento della attività di testing si basa sulla identificazione dei cosiddetti **casi di test**. Un caso di test è un insieme di dati forniti in ingresso ad un programma allo scopo di verificare la corrispondenza dei risultati ottenuti con quanto previsto dalle specifiche.

Normalmente, un caso di test è insufficiente a verificare la qualità di un programma, il problema è identificare un **insieme di casi di test** "appropriato".

Il termine "appropriato" rimanda ai due principali aspetti critici dell'attività di test, in primo luogo, il costo dell'attività di test è proporzionale alla qualità del risultato che si vuole ottenere; è necessario modulare il volume di test che si decide di effettuare in funzione del costo di tale attività alla luce dei requisiti del programma e dell'utente, per esempio un software per la gestione di un sistema di avionica richiederà un'attività di testing diversa da quanto necessario per un video-game.

Il secondo punto critico è la necessità di identificare casi di test non ridondanti; due casi di test possono essere equivalenti, in quanto stimolano il programma allo stesso modo ed allora in questo caso diviene inutile includerli entrambi nell'insieme di casi di test prescelto.

E' necessario, inoltre, identificare il contesto operativo all'interno del quale viene effettuata l'attività di test e poi procedere con un testing opportuno.

I tipi di test di un programma sono classificabili secondo i seguenti livelli:

- Test di unità: mirato alla correttezza degli algoritmi per cui si verifica singolarmente ogni modulo del programma;
- Test di integrazione: mirato alla correttezza delle interfacce tra moduli, si verifica l'integrazione tra un subset dei moduli.
- Test di sistema: una volta realizzati i vari moduli, questi vanno assemblati e quindi va verificato il comportamento del sistema nel suo complesso considerando l'aspetto di affidabilità, sicurezza e prestazioni;
- Test di accettazione: viene imposto dal cliente e verifica il rispetto dei requisiti;
- Test di regressione: verifica che una nuova versione di un programma superi tutti i test già effettuati su versioni precedenti

dello stesso per controllare che non siano introdotti errori in versioni successive.

Tipicamente si ha un primo testing effettuato dagli sviluppatori una volta ultimato il programma, quindi si passa ad una fase di **testing interno**.

Il testing interno, detto anche **α -test**, viene normalmente effettuato da persone diverse dagli sviluppatori del sistema, ma ancora interne alla ditta produttrice del software, una volta terminata la fase di α -test viene rilasciata una versione non definitiva del prodotto che viene proposta per una fase di **β -test (test esterno)** da parte di veri utenti (detti anche **utenti pilota**).

3 IL COLLAUDO

Il Collaudo ha un connotato di validazione e viene eseguito dopo il completamento dei test, ed è orientato all'accettazione formale dei sistemi.

La fase di Collaudo, che operativamente conclude il Ciclo di Sviluppo del Software, si basa su alcune specifiche pianificazioni ed attività:

- pianificazione delle attività di Prova;
- attività di Allestimento delle Prove;
- attività di Esecuzione delle Procedure di Prova e Valutazione dei Risultati di Prova (Collaudo).

Nel seguito si definiscono le modalità di effettuazione di ogni singola attività.

Pianificazione delle attività di prova

Il costo di una estesa attività di collaudo non è giustificata per tutti i prodotti software, quindi, la prima cosa che deve essere definita è la strategia che si vuole perseguire nel collaudo di un prodotto software in base ai Requisiti specificati e tenendo conto delle risorse a disposizione. Queste indicazioni di pianificazione devono essere riportate nel Piano di Collaudo.

I contenuti del piano sono definiti sulla base dei requisiti di qualità riportati nelle Specifiche dei Requisiti del prodotto software da collaudare.

Il Piano di Collaudo fornirà le seguenti indicazioni:

- i requisiti generali dell'attività di collaudo e l'identificazione di ogni singola prova;
- a quali livelli di integrazione si intende collaudare ogni singolo requisito del prodotto software;
- le tecniche e gli strumenti di collaudo che si vogliono utilizzare nelle attività di preparazione ed esecuzione delle prove; in particolare si indicheranno le tecniche di definizione dei casi di prova che si intendono utilizzare;

- indicazione di massima degli attributi che dovranno essere posseduti dalle prove (classi di prova);
- l'ambiente hardware e software che dovrà essere utilizzato per l'allestimento e l'esecuzione delle prove;
- gli strumenti di supporto utilizzati, sia in ambiente host di sviluppo che in ambiente target, per l'allestimento e l'esecuzione delle prove, nonché gli strumenti utilizzati per la registrazione e la valutazione dei risultati ottenuti e del grado di copertura topologica raggiunta;
- le risorse che vengono assegnate per l'allestimento e l'esecuzione delle attività di prova

Allestimento delle prove

La normale attività di allestimento di ogni singola prova identificata nel Piano di Collaudo, avviene attraverso le seguenti due attività elementari:

1. **Functional test cases** (definizione dei casi di prova funzionali).
In base alle prove identificate nel Piano di Collaudo ed ai requisiti funzionali da provare, vengono selezionati i dati funzionali da sottoporre a prova, i risultati di prova attesi ed il criterio di valutazione dei risultati reali; l'insieme di queste informazioni costituiscono i functional test cases di una singola prova.
2. **Procedure di prova.**
Per ogni test case deve essere definita una procedura (automatica e/o manuale) in grado di provare, attraverso esecuzione del codice "as-built" ed ai livelli di integrazione hw/sw previsti, la corretta implementazione di un singolo requisito.

Tra gli obiettivi che ci si deve porre nell'allestimento delle prove sono di considerevole rilievo:

- la ripetibilità delle prove;
- l'affidabilità ed oggettività delle prove e del criterio di valutazione dei risultati;
- l'economicità dell'attività di collaudo;
- la rapidità con cui si riescono ad evidenziare le anomalie.

Per tali ragioni è utile, ogniqualvolta sia possibile, realizzare le prove attraverso l'esecuzione di procedure automatiche con l'obiettivo di ridurre al minimo l'intervento umano durante l'attività di collaudo e registrazione dei risultati.

Collaudo

La fase di Collaudo del Ciclo di Sviluppo di un prodotto software viene svolta tramite l'esecuzione delle procedure di prova sugli elementi software sviluppati.

Il Collaudo si compone sostanzialmente di due attività elementari:

1. **l'esecuzione delle prove;**
2. **la segnalazione di anomalie software.**

L'esecuzione delle prove avviene preparando l'ambiente hardware e software descritto, attivando l'oggetto da collaudare ed i simulatori eventualmente necessari, sottoponendo l'oggetto da collaudare ai passi previsti dalla procedura di prova, registrando gli esiti e confrontando quelli ottenuti con quelli previsti.

In ogni caso, gli esiti di tutti i test cases previsti per ogni singola prova eseguita sono raccolti nel documento **Rapporto Verifica Progetto Software**.

In tale documento per ogni singola prova deve essere indicato:

- l'elemento software sottoposto al collaudo;
- l'identificazione della prova eseguita;
- l'ambiente di prova adottato;
- l'esito sintetico dell'intera prova e di tutti i test cases che la compongono;
- l'identificazione della Segnalazione Anomalia Software associata ad ogni prova fallita.
- Segnalazione anomalie software

Per ogni prova che fornisca un esito negativo deve essere compilata una Segnalazione Anomalia Software identificata anche nel Rapporto Verifica Progetto Software.

La **segnalazione di anomalia software** comporta l'aggiornamento controllato della Configurazione Interna di Sviluppo e la riconvocazione dei riesami tecnici che si dovessero rendere necessari per la ri-approvazione degli elementi progettuali aggiornati. Tale sequenza di attività viene ripetuta ciclicamente sino a quando tutte le prove eseguite non diano esito positivo.

Conclusa l'attività di Collaudo, va redatto il **Verbale di Collaudo**.

Si tratta del documento che esprime il giudizio complessivo sul collaudo del prodotto in termini di corrispondenza con i requisiti richiesti. In questo documento oltre che a mettere in luce i problemi trovati e evidenziare soluzioni ipotizzabili è possibile raccomandare la riesecuzione del collaudo per quegli aspetti non soddisfacenti i requisiti richiesti e per altri che potrebbero venire influenzati dalle modifiche apportate per risolverli.

Questo rappresenta il documento base per definire insieme al team di progetto un piano d'azione per risolvere le non conformità rilevate.

Lo schema del verbale di collaudo è il seguente:

- Sintesi per il management (Breve sunto con considerazioni essenziali, conclusioni e raccomandazioni più importanti)
- Descrizione del collaudo (descrizione delle condizioni di test, degli strumenti, dei metodi, delle attività e di tutto ciò possa essere utile per identificare i test effettuati)

- Rilevazioni (dati scaturiti dal collaudo)
- Conclusioni (valutazione dell'analisi dei dati, identificazione dei problemi, del loro impatto sul sistema)
- Raccomandazioni (suggerimenti riguardo le azioni da intraprendere per ridurre o eliminare i problemi riscontrati)

Nel caso in cui la fase di collaudo avesse evidenziato gravi difformità rispetto ai requisiti richiesti è opportuno ritornare alla fase in cui è possibile ipotizzare siano stati commessi gli errori rilevati; qualora questi risultino essere di tipo architettonico o progettuale si rende necessario il ritorno alla corrispondente fase e riprogettare l'applicazione.

Da notare come casi di questo tipo comportino costi enormi di rework del progetto; in ogni caso occorre revisionare il piano di progetto aggiornandolo sui tempi previsti per la rilavorazione e a fine della quale rieffettuare per intero la fase di collaudo.

In questi casi è consigliabile una revisione dell'intero ciclo di vita, ripensando la strutturazione del processo produttivo stesso.

Nei casi in cui si sono accertate delle difformità dai requisiti richiesti, che dai verbali redatti dal team di collaudo sono giudicate risolvibili in tempi brevi e soprattutto con impatto limitato sul prodotto, allora questo significa che dopo la fase di correzione non si rende indispensabile la ripetizione di tutta la fase di collaudo ma solo di una parte limitata di essa connessa alle modifiche apportate.

L'accorgimento da adottare in fase di ripetizione del collaudo dopo le modifiche, è quello di sottoporre a test di regressione le parti o le caratteristiche di prodotto che avrebbero potuto in qualche modo essere modificate dalle correzioni effettuate.

Una volta che il prodotto sia risultato conforme ai requisiti di progetto viene redatto dal team di collaudo il **Verbale di Accettazione / Validazione**

Questo verbale costituisce l'unico documento in grado di validare il prodotto (se questo è stato sviluppato all'interno) o di accettarlo (nel caso sia stato sviluppato da un fornitore esterno).