Sommario

Introduzione 3

Il linguaggio 3

I tipi di dato 4

Booleani 4

Numeri 4

Caratteri e stringhe 4

Linguaggi informatici 5

Interprete/compilatore 5

Tipi di errore 5

Lo sviluppo di un software 6

Analisi dei requisiti 6

Analisi 6

Progettazione 6

Realizzazione 6

Validazione 6

Rilascio e documentazione 7

Cicli di vita dei software 8

Modello a cascata 8

Modello evolutivo/incrementale 8

Modello a spirale 9

La qualità del software 10

ISO/IEC 9216: cos'è? 10

Organizzazione internazionale per la normazione (ISO) 10

Commisione elettrotecnica internazionale (IEC) 10

La ISO/IEC 9126 10

Modello della qualità del software 11

Metriche per la qualità esterna 12

Metriche per la qualità interna 13

Metriche per la qualità in uso 14

Il costo del software 15

Diagramma di Gantt 15

Vantaggi di un diagramma di Gantt 16

Chiarezza 16

Comunicazione 16
Motivazione 16
Gestione del tempo 16
Flessibilità 16
Svantaggi dei diagrammi di Gantt 17
La complessità 17
Linearità 17
Lavoro impegnativo 17

Introduzione

L'informazione è tutto ciò che possiede un significato per l'uomo, e che viene conservato o utilizzato in vista di una utilità pratica seguente.
Colui che trasmette l'informazione viene definito l'**emittente**, chi riceve l'**informazione** si chiama il ricevente. Il mezzo che trasporta l'informazione viene nominato genericamente **canale** e viene tradotto per mezzo di un codice di **segnali**.

Il linguaggio

Il linguaggio, come si intende in senso generale, è lo strumento stesso con cui si attua la comunicazione. Esso è una facoltà propria dell'uomo di esprimersi e comunicare tramite un sistema di simboli, in particolare di segni vocali e grafici: *l. verbale, non verbale.*

I linguaggi usati nella trasmissione di informazioni si distinguono in linguaggi naturali e linguaggi formali.

I linguaggi **naturali** sono i linguaggi che gli uomini usano normalmente per comunicare ma a volte possono dare origine ad ambiguità, come nel seguente esempio :

La giovane mente

Può essere interpretata in 2 modi:

a) una ragazza mente.

b) una persona di cui la mente è giovane.

I linguaggi **formali** sono basati su simboli e regole ben precise, **non danno origine ad ambiguità** e sono dedicati a scopi ben precisi: es nell'ambito della matematica, linguaggi formali sono i simboli e le regole dell'insiemistica o della geometria.

Ogni linguaggio ha i suoi termini e le sue regole ma ci sono alcune caratteristiche comuni dei linguaggi formali: alfabeto, sintassi, semantica.

Innanzitutto un linguaggio è costruito su un **alfabeto** di **simboli convenzionali** detti **caratteri**. Se i caratteri dell'alfabeto sono lettere possiamo formare un insieme di sequenze dette **parole**. Nel caso invece del linguaggio della logica vengono usate le lettere dell'alfabeto per indicare le proposizioni mentre i simboli ¬, →, ^, per indicare i connettivi.

Una sequenza di caratteri dell'alfabeto costituisce una stringa (per es il numero 3215 è una stringa dell'alfabeto dell'aritmetica mentre a v b è una stringa dell'alfabeto della logica).

Ci sono stringhe che hanno significato in un determinato linguaggio mentre altre non ne hanno.

Semantica: è l'insieme dei significati che devono essere attribuiti alle stringhe.

Sintassi: è l'insieme delle regole che debbono essere applicate per costruire una stringa che abbia significato. Quindi la sintassi ci permette di stabilire se una stringa appartiene o meno al linguaggio considerato.

L'insieme di tutte le regole che consentono di generare parole e frasi del linguaggio considerato si dice **grammatica**.

I tipi di dato

I dati sono delle conoscenze elementari che caratterizzano una situazione reale e che possono essere conservate per un futuro utilizzo. Essi quindi possono essere considerati come degli elementi costitutivi delle informazioni. Il trattamento dei dati per ottenere nuove informazioni viene detto elaborazione. Ciascuna elaborazione utilizza dei dati in ingresso, detti dati di input, e produce dei dati in uscita, detti dati di output. I dati possono essere di tipo numerico, interi o double, di tipo alfanumerico, char o string, o che riguardano l'algebra booleana.

Booleani

Il tipo booleano prevede due soli valori: true ("vero") e false ("falso"). Questi valori vengono utilizzati in modo speciale nelle espressioni condizionali per controllare il flusso di esecuzione; inoltre possono essere manipolati con gli operatori booleani AND, OR, NOT e così via.

Anche se in teoria basterebbe un solo bit per memorizzare un valore booleano, per motivi di efficienza si usa in genere un'intera parola di memoria, come per i numeri interi "piccoli" (una parola di memoria a 8 bit, per esempio, può memorizzare numeri da 0 a 255, ma il tipo booleano utilizza solo i valori 0 e 1).

Numeri

I tipi di dati numerici includono i numeri interi e i numeri razionali in virgola mobile, che sono astrazioni dei corrispondenti insiemi di numeri della matematica. Quasi tutti i linguaggi includono tipi di dati numerici come tipi predefiniti e forniscono un certo numero di operatori aritmetici e di confronto su di essi.

A differenza degli insiemi numerici della matematica, i tipi di dati numerici sono spesso limitati (includono cioè un massimo e un minimo numero rappresentabile), dovendo essere contenuti in una singola parola (word) di memoria.

Caratteri e stringhe

Il tipo carattere contiene, per l'appunto, un carattere: generalmente si riferisce ad un carattere ASCII e viene memorizzato in un byte. Tuttavia in questi anni si sta affermando il nuovo standard Unicode per i caratteri, che prevede 16 bit (che generalmente corrisponde a una parola di memoria) per la rappresentazione di un singolo carattere. Molti linguaggi tradizionali si sono adattati a questo standard emergente introducendo, in aggiunta al tipo "carattere a 8 bit", un nuovo tipo "carattere a 16 bit", talvolta detto wide char (il linguaggio Java è invece un esempio di linguaggio moderno che gestisce direttamente tutti i caratteri nel formato Unicode).

Le stringhe sono sequenze di caratteri di lunghezza finita. I linguaggi possono fornire operazioni per la concatenazione di stringhe, la selezione di sottostringhe di una stringa data, ecc.

Date e orari

Con il formato Datetime vengono rappresentati dei dati con sintassi dd/mm/yyyy hh:mm:ss. Questo significa che le date possono essere rappresentate separando giorni, mesi e anni con il carattere '**/**', mentre gli orari separando con '**:**' le ore dai minuti e dai secondi. Ad esempio, mezzogiorno e un quarto del 3 gennaio 2030 sarà rappresentato in questo modo : **03/01/2030 12:15:00**.

Linguaggi informatici

Per "comunicare" con un computer, il linguaggio deve essere tradotto in un codice che il dispositivo può capire.

Il linguaggio usato per scrivere programmi per computer è detto linguaggio di programmazione (o linguaggio di alto livello), ad esempio C++, HTML, Java.

Il testo di un programma è detto codice, perché è il risultato della traduzione dell'algoritmo in una forma diversa e comprensibile al computer.

Il linguaggio macchina (detto anche linguaggio di basso livello) è, invece, quello utilizzato dal computer al suo interno ed è composto da istruzioni elementari inviate al processore, è formato da 0 e 1.

Interprete/compilatore

Ha la funzione di "tradurre" il codice sorgente, scritto in linguaggio ad alto livello, in codice oggetto, scritto in linguaggio di basso livello. Durante la traduzione non avviene nessun cambiamento, infatti, il codice sorgente è uguale a quello oggetto per significato.

Tipi di errore

Gli errori possono essere di 2 tipi:

- Errori formali: lessicali e sintattici, contenuti nel programma sorgente.
- Errori logici: dovuti a un algoritmo sbagliato, rilevabili solo con l'esecuzione del programma, perché producono risultati diversi da quelli attesi. Sono conosciuti anche come **Bug**.

Lo sviluppo di un software

Analisi dei requisiti

L'analisi dei requisiti è un'attività che precede lo sviluppo (o la modifica) di un sistema software, il cui scopo è quello di definire le funzionalità che il nuovo prodotto (o il prodotto modificato) deve offrire, cioè i requisiti che devono essere soddisfatti dal software sviluppato. L'analisi dei requisiti è una fase presente in tutti i modelli di ciclo di vita del software. Nel modello a cascata, l'analisi dei requisiti è la prima fase del processo di sviluppo, e deve concludersi con la stesura di una dettagliata specifica dei requisiti che descrive tutte le funzionalità del nuovo software; in base a questa specifica verranno impostate le fasi successive di sviluppo.

Analisi

l'analisi è una delle fasi del ciclo di vita del software; ha lo scopo di chiarire, dettagliare e documentare le funzioni, i servizi e le prestazioni che debbono essere offerti da un software o programma, per risolvere un dato problema nel contesto in cui dovrà operare. Le informazioni raccolte nella fase di analisi rappresentano il punto di partenza per la progettazione di un software e per l'intero processo della sua realizzazione, validazione e manutenzione.

In particolare vengono individuati i dati e le funzioni utilizzate dal software.

Progettazione

L'obiettivo di questa fase è quello di definire e descrivere dettagliatamente le caratteristiche del sistema informatico del progetto.

La progettazione produce documenti chiari e non ambigui, che devono essere seguiti fedelmente dagli sviluppatori. Nello specifico, un documento di progetto potrà contenere una descrizione mediante **metacodice (o metalinguaggio)** oppure mediante **diagrammi di flusso**.

Realizzazione

Dopo lo studio, l'ideazione e la concretizzazione delle scelte di progetto, si arriva alla realizzazione del prodotto del progetto. Trattandosi di un progetto informatico il prodotto è costituito da moduli software, detti **programmi**, e dagli archivi elettronici utilizzati dai programmi stesso.

Validazione

Il programma creato viene sottoposto ad una serie di test per verificare la correttezza di tutte le funzionalità del software. I test saranno prodotti prima, e in modo indipendente, della fase di realizzazione, e di solito da figure professionali differenti.

Rilascio e documentazione

Se il programma in fase di validazione passa tutti test, lo si può rilasciare, e vengono redatti tutti i documenti di supporto, da poter consultare in futuro nel caso si debbano apportare degli aggiornamenti o modifiche al programma.

Possono venire redatte anche le guide per l'utente, che spiegano all'utilizzatore come usare le varie funzionalità del prodotto, ed eventualmente anche dei documenti in cui si attesta il superamento dei test.

Cicli di vita dei software

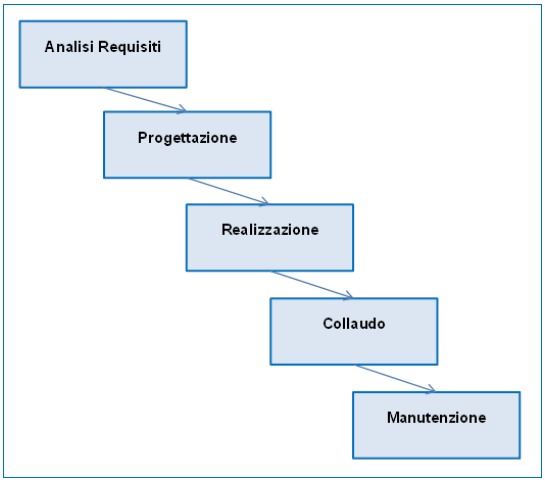
Il ciclo di vita del software si riferisce al modo in cui una metodologia di sviluppo suddivide il processo di realizzazione di un software ottimizzando cosi le diverse fasi che sono a loro volta coordinate:

- analisi
- progettazione
- sviluppo
- collaudo
- manutenzione

Principali esempi di metodologia di sviluppo

Modello a cascata

Il modello Waterfall è stato inventato da Winston Royce nel 1970 e viene chiamato così perché ogni fase del ciclo di vita del software si sussegue a sequenza facendo progredire lo sviluppo proprio come una cascata. In questo modello ogni fase produce un output che verrà usato come imput nella fase successiva ed ogni fase del processo sarà documentata.

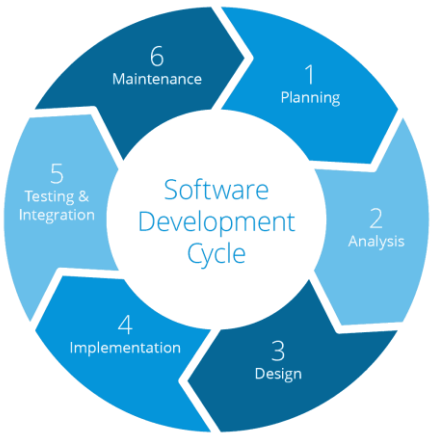


Questo metodo, però, ha dei limiti:

- Assenza di flessibilità: se, dopo un po' di tempo, il cliente ha bisogno di altri requisiti, il software deve essere prodotto da capo.
- L'aggiunta di nuove funzionalità richiede tempi elevati.
- Aumento considerevole dei costi.
- Linearità: spesso si hanno dei cicli di feedback per la correzione degli errori.

Modello evolutivo/incrementale

Il modello evolutivo, successivamente diventato modello incrementale, cerca di superare i limiti principali del modello a cascata riuscendoci parzialmente, questo sistema si evolve mano a mano che i requisiti vengono compresi, eliminando così gli errori di incomprensione ma non riducendo le tempistiche.



Questo modello nasce grazie a 17 ingegneri che ogni giorno si riunivano per discutere sulle problematiche della creazione di un software. Esso contiene una fase in più rispetto agli altri modelli: la

valutazione. Questo ciclo viene ripetuto fino a che la valutazione del prodotto diviene soddisfacente rispetto ai requisiti richiesti.

Il fallimento dei vari prototipi porta a una modifica dei requisiti o a una revisione del progetto riducendo così volta per volta il rischio di fallimento, e questo processo viene ripetuto fino a quando il software non si reputa soddisfacente in base ai requisiti del cliente

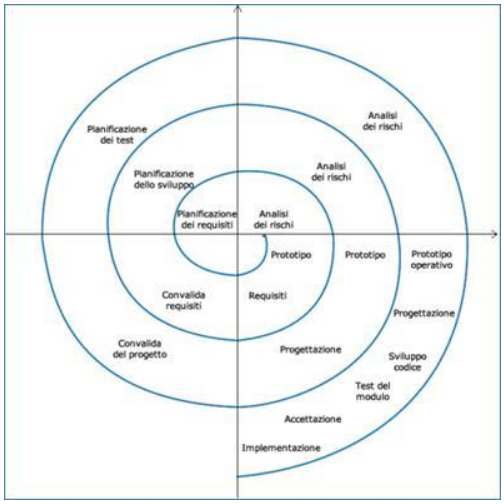
L'obiettivo di questo software è di rispondere ai cambiamenti invece di seguire un piano, di migliorare sempre di più, in modo tale da soddisfare i requisiti del cliente anche in futuro.

Il modello incrementale sta diventando il più adottato rispetto agli altri modelli, quindi, sembra proprio che le parole chiave di uno sviluppo sempre più efficiente nel campo dei software siano flessibilità e cambiamento.

Modello a spirale

Questo modello viene inventato da Barry Boehm e nasce con l'obiettivo di superare il modello "a cascata". Esso punta ad avere una maggiore flessibilità rispetto al modello Waterfall. Le fasi vengono scomposte i quattro fasi multiple, ciascuna ripetuta più volte.

Quindi, le fasi restano le stesse ma i tempi sono più ristretti e dalla fase di verifica si torna a quella di analisi per eventuali correzioni.



Ripete in modo iterativo quattro fasi, chiamate spirali:

- identificazione dei requisiti
- progettazione
- produzione
- analisi dei rischi

Questo modello al contrario di quello a cascata è molto flessibile e permette di aggiungere elementi mano a mano che si presentano, derivanti ad esempio da un cambiamento di requisiti del cliente, riducendo così anche i tempi di realizzazione.

Anche se è considerato uno dei migliori modelli attualmente applicabili presenta comunque dei difetti, ad esempio costi elevati per un eventuale miglioramento o la possibilità di avere una documentazione estremamente complessa. In più non è adatto a progetti di piccole dimensioni, e risulta difficile la definizione delle fine del progetto stesso.

La qualità del software

Come scegliere il giusto software?

Per scegliere il giusto software è molto importante fare un'analisi della sua qualità, ciò significa analizzare quante necessità del compratore soddisfa, in base a com'è composto al suo interno. La qualità di un software deve rispettare dei determinati parametri, sia esterni che interni, che sono stati scelti ed analizzati dalle ricerche nel campo dell'ingegneria del software.

ISO/IEC 9216: cos'è?

Si parla della norma ISO/IEC 9126 quando si vogliono raggruppare tutti i parametri per definire la qualità di un software. ISO e IEC sono le abbreviazioni dell'Organizzazione internazionale per la normazione e la Commissione elettrotecnica internazionale.

Organizzazione internazionale per la normazione (ISO)

L'Organizzazione internazionale per la normazione, acronimo di ISO, è la più importante organizzazione a livello mondiale per la definizione di norme tecniche. L'ISO fu fondata il 23 febbraio 1947 in seguito ad un incontro all'interno dell'Institute of Civil Engineers a Londra, avvenuto nel 1946, al quale parteciparono delegati provenienti da 25 stati diversi. Nell'anno della sua fondazione, l'ISO contava 67 comitati tecnici e fu una delle prime organizzazioni chiamate alla consultazione dal Consiglio economico e sociale delle Nazioni Unite. Ha la sua sede a Ginevra in Svizzera, e i suoi membri sono gli organismi nazionali di standarizzazione di 164 paesi del mondo. Dalla sua nascita fino al 2019, l'ISO ha sviluppato 22.683 norme tecniche internazionali.

Commisione elettrotecnica internazionale (IEC)

La Commissione elettrotecnica internazionale**,** acronimo di IEC è un'organizzazione internazionale per la definizione di standar in materia di elettricità, elettronica e tecnologie correlate. Questa commissione è formata da rappresentanti di enti di standardizzazione nazionali riconosciuti. La IEC è stata fondata nel 1906 ed inizialmente aveva sede a Londra; nel 1948 ha spostato la sua sede a Ginevra a cui attualmente partecipano più di 60 paesi. La IEC ha il compito di sviluppare e distribuire gli standard per le unità di misura, in particolare il *gauss*, *l'hertz* e il *weber*; inoltre, fu la prima a proporre un sistema di standard, il Sistema Giorgi, che intorno al 1960 si è modificato nel sistema delle unità SI (sistema internazionale di unità di misura).

La ISO/IEC 9126

La sigla ISO/IEC 9126 comprende una serie di norme e linee guida contenenti il modello di riferimento per definire le caratteristiche di qualità del software per fare in un modo che le società di software possano migliorare l'organizzazione e i processi, quindi, come conseguenza, la qualità del prodotto sviluppato e le metriche per la valutazione della qualità che il software possiede. Per qualità del software si intende la misura in cui un software soddisfa un certo numero di aspettative rispetto al suo finanziamento e alla sua struttura interna.

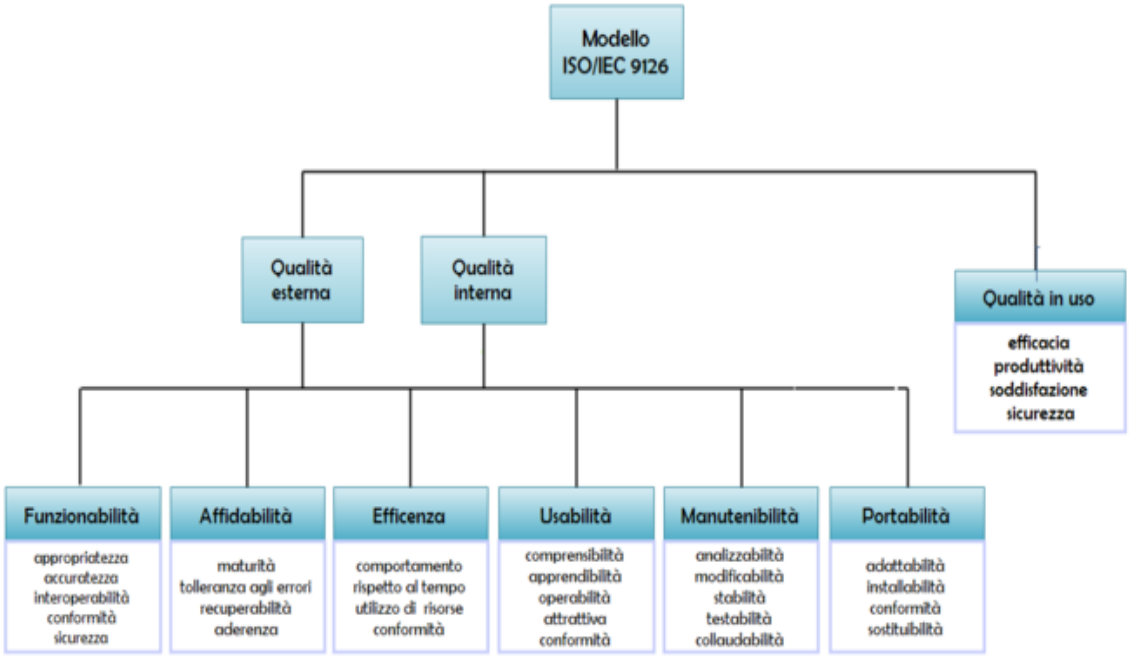
La norma tecnica relativa alla qualità del software si divide in quattro parti:

1. Modello della *qualità del software*
2. Metriche per la *qualità esterna*
3. Metriche per la *qualità interna*
4. Metriche per la *qualità in uso*

Modello della qualità del software

Il modello di qualità stabilito nella prima parte dello standard, è classificato da sei caratteristiche fondamentali; funzionalità, che è un "requisito funzionale", a differenza delle altre cinque caratteristiche, ovvero, affidabilità, efficienza, usabilità, manutenibilità e portabilità che sono chiamate invece "requisiti non funzionali" o "requisiti di qualità".

- la ***funzionalità*** è la capacità di un prodotto software di fornire funzioni che soddisfano le esigenze necessarie per operare a delle condizioni specifiche.
- l'***affidabilità*** è la capacità del prodotto software di mantenere uno specifico livello di prestazioni quando usato in date condizioni per un dato periodo.
- l'***efficienza*** è la capacità di fornire appropriate prestazioni relativamente alla quantità di risorse usate.
- l' ***usabilità*** è la capacità del prodotto software di essere capito e usato dall'utente senza nessun problema anche quando è usato sotto condizioni specifiche.
- la ***manutenibilità*** è la capacità del software di essere modificato, includendo correzioni, miglioramenti o adattamenti; la manutenibilità può essere correttiva (corregge gli errori dei clienti) o adattativa (si adatta alle nuove richieste del cliente).
- la ***portabilità*** è la capacità del software di essere trasportato da un dispositivo ad un altro.



Metriche per la qualità esterna

Le metriche esterne, misurano i comportamenti del software sulla base dei test e dall'osservazione durante la sua esecuzione, e dalle funzionalità che offre in funzione degli obiettivi stabiliti.

Questi parametri misurano il comportamento del software a confronto con svariati test durante esecuzione.

1. Correttezza o funzionalità, cioè se soddisfa pienamente le richieste e se fa ciò che è stato programmato di fare. Questo parametro deve rispettare:
   a. Appropriatezza, cioè la capacità del software di dare tante funzioni quante le necessità dell'utente.
   b. Accuratezza, cioè la capacità del software di garantire un risultato.
   c. Interoperabilità, cioè la capacità del software di funzionare su piattaforme diverse.
   d. Conformità, cioè la capacità di rispettare gli standard, le convenzioni e le regole del settore in cui agisce.
   e. Sicurezza, cioè la capacità di proteggere dei dati.
2. Affidabilità, cioè, in modo molto generale, è quanto l'utente si può effettivamente fidare del software. Nello specifico, secondo la definizione della Mean time between failure, un software diventa più affidabile se i suoi malfunzionamenti sono molto rari o completamente assenti.
3. Efficienza, cioè se usa la memoria del dispositivo senza sprecarla; in parole povere se il software usa il minor numero possibile di istruzioni e allo stesso tempo usando correttamente lo sviluppo. L'efficienza molto spesso è un parametro talmente importante che spesso viene identificato come requisito dall'utente. Un esempio di efficienza potrebbe essere la velocità di avvio di un programma.
4. Usabilità, cioè rendere accessibile il software a chiunque, rispettando:
   a. Comprensibilità, cioè l'esprimere con facilità lo scopo, le caratteristiche e le funzioni del sofrtware.

   b. Apprendibilità, cioè la capacità rendere il programma talmente semplice da ridurre al minimo lo sforzo dell'utente nel comprendere l'uso del software.
   c. Attrattiva, cioè principalmente la presentazione estetica del prodotto.
5. Ecocompatibilità, cioè l'impatto dell'uso del software sull'ambiente.
6. Scalabilità, cioè l'adattazione del software, senza doverlo riprogrammare da capo.

Metriche per la qualità interna

La qualità interna, (più precisamente le metriche interne), si applica al software non eseguibile durante le fasi di progettazione e codifica. Le metriche interne permettono di individuare eventuali problemi che potrebbero influire sulla qualità finale del prodotto prima che sia realizzato il software eseguibile. Esistono metriche che possono simulare il comportamento del prodotto finale tramite simulazioni.

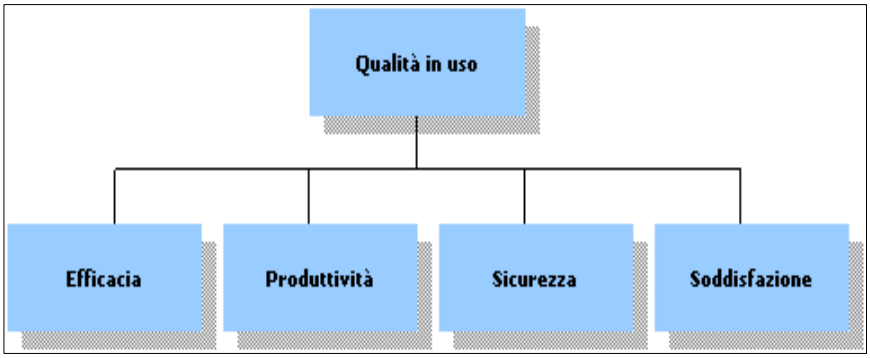
Questi parametri misurano il software nelle fasi di progettazione e programmazione.

1. Verificabilità, cioè se i due parametri esterni quali correttezza ed affidabilità sono facilmente verificabili.
2. Manutenibilità, cioè l'apporto di modifiche di tipo correttive (eliminando errori già esistenti o causati da precedente manutenzione), adattive (modifiche per il parametro dell'ecocompatibilità e modifiche dell'hardware e/o del sistema operativo) o evolutive (qualità e nuove funzionalità).
3. Riusabilità, cioè il riutilizzo di vecchie parti di software per crearne di nuovi.
4. Portabilità, cioè la capacità di adattarsi su diverse piattaforme; come ad esempio un gioco che funziona su un su diversi sistemi operativi degli smartphone, quali Android, IOS o Windows.

Metriche per la qualità in uso

La qualità in uso rappresenta il punto di vista dell'utente sul software. Il livello di qualità in uso è raggiunto quando sia stato raggiunto sia il livello di qualità esterna sia il livello di qualità interna. La qualità in uso, quindi, permette di abilitare specificati utenti ad ottenere specificati obiettivi con *efficacia*, *produttività*, *soddisfazione* e *sicurezza.*

- *Efficacia*: la capacità del software di mettere in grado gli utenti di raggiungere gli obiettivi specificati con accuratezza e completezza.
- *Produttività*: la capacità di mettere in grado gli utenti di spendere una quantità di risorse appropriate in relazione all'efficacia ottenuta in uno specificato contesto d'uso.
- *Sicurezza*: la capacità del prodotto software di raggiungere accettabili livelli di rischio di danni a persone, al software, ad apparecchiature o all'ambiente operativo d'uso.
- *Soddisfazione*: è la capacità del prodotto software di soddisfare gli utenti.



Il costo del software

Le **funzioni** (o moduli) ossia le cose che posso fare, mentre l'altra parte è costituita dai **servizi**. In altre parole, la formazione del personale per imparare ad utilizzare il software, l'inserimento dei dati per poter adattare il software alle esigenze specifiche della nostra azienda e infine, in presenza di macchinari di produzione, anche delle eventuali verifiche in macchina per certificare la correttezza dei progetti, prima di procedere con la produzione in serie. Oltre al prezzo di acquisto, esistono poi una serie di **costi "nascosti",** come il costo legato all'apprendimento del software. Sebbene normalmente nel prezzo di acquisto può essere inserito un corso di formazione. Infatti una volta che il corso con il tecnico dell'azienda fornitrice sarà terminato, inevitabilmente sarà necessario altro tempo prima di poter veramente padroneggiare il software in modo corretto. E naturalmente questo tempo aggiuntivo altro non è, che ulteriori soldi. Ma la cosa a cui si deve fare molta attenzione è alla semplicità che un software dimostra di avere, attraverso la sua **interfaccia utente**. Quindi se viene acquistato un software con un'interfaccia semplice, dove grazie alla pressione di pochi tasti intuitivi si ottengono i risultati, anche il tempo di apprendimento sarà inferiore e di conseguenza i costi saranno inferiori. Se invece viene acquistato uno strumento, magari più economico che però ha interfacce di oltre dieci anni fa, prevalentemente testuali, poco intuitive e macchinose, i costi aumenteranno perché nel caso di un'azienda, serviranno più ore per i collaboratori per diventare autonomi.

Diagramma di Gantt

Il diagramma di Gantt è uno strumento utile per la pianificazione dei progetti. Attraverso una panoramica dei compiti programmati, tutte le parti interessate sono a conoscenza dei compiti e delle rispettive scadenze.

Un diagramma di Gantt mostra:

1. Le date di inizio e fine di un progetto
2. Da quali attività è composto il progetto
3. Le attività assegnate a ciascuna persona
4. Le date previste per l'inizio e la fine delle attività
5. Una stima di quanto tempo durerà ogni attività
6. Come le attività si sovrappongono e/o sono collegate tra loro

I progetti consistono in una serie di sotto-attività. Con un diagramma di Gantt è possibile tenere traccia di queste ultime, in modo da non dimenticare o dover posticipare nulla. Il grafico indica quando ogni attività dovrebbe essere realizzata. Questo aiuta a garantire che ogni sotto-attività e di conseguenza l'intero progetto sia portato a termine nelle tempistiche previste. Alcune attività possono essere eseguite in qualsiasi momento, mentre altre devono essere completate prima o dopo l'inizio o la fine di un'altra attività. Un diagramma di Gantt mostra tali dipendenze. Il grafico ti mostra l'esatto andamento del progetto e ti fa vedere quali attività sono già state eseguite. Grazie all'indicazione della data, avrai una panoramica di quanto ti resta da fare, per renderti conto se sei sulla buona strada per terminare in tempo il progetto.

Vantaggi di un diagramma di Gantt

L'uso di un diagramma di Gantt nel processo di gestione del progetto offre i seguenti vantaggi:

1. Chiarezza
2. Panoramica semplificata
3. Approfondimenti sulle prestazioni
4. Migliore gestione del tempo
5. Flessibilità

Chiarezza

Un diagramma di Gantt suddivide più attività e timeline in un'unica panoramica. In questo modo saprai a che punto si trova il tuo team in ciascuna della fasi di un progetto. Utilizza il grafico per mostrare agli stakeholder come sono organizzati i compiti e quali risorse vengono assegnate. Potrai così fornire loro un quadro chiaro dell'andamento del progetto e mostrargli se sarà completato entro i termini accordati.

Comunicazione

I team di lavoro utilizzano diagrammi di Gantt per sostituire le riunioni e fornire aggiornamenti sullo stato di avanzamento del progetto, aiutando i membri del gruppo ad avere una chiara visione d'insieme dei progressi delle attività.

Motivazione

Grazie a una visione d'insieme chiara, i membri del team possono comprendere meglio le loro prestazioni complessive e sono in grado di inserire le proprie abitudini di lavoro nella pianificazione del progetto. Saranno inoltre più coscienti delle dipendenze tra i compiti e avranno una migliore consapevolezza dell'impatto dei ritardi sul progetto. Ciò favorisce una collaborazione più stretta e incentiva una migliore organizzazione dei compiti.

Gestione del tempo

I diagrammi di Gantt ti aiutano a impostare tempi realistici. Le barre sul grafico indicano entro quale intervallo di tempo deve essere completato un determinato compito. In questo modo, puoi anche assicurarti che altri progetti che consumano risorse e tempo non siano compromessi.

Flessibilità

I progetti sono spesso soggetti a cambiamenti. Avendo a disposizione una panoramica dei cambiamenti inattesi a livello di attività o di tempistiche, è possibile adattare di conseguenza le attività e le risorse.

Svantaggi dei diagrammi di Gantt

La complessità

I progetti composti da un elevato numero di attività e risorse possono dare come risultato diagrammi di Gantt molto complessi, specialmente se si lavora su un progetto esteso o se il team coinvolto è composto da numerosi membri. Visualizzare decine, o anche centinaia di attività su un diagramma di Gantt potrebbe rivelarsi un'ardua impresa. Pertanto, è fondamentale pensare in anticipo a diversi compiti e sottocompiti e designare qualcuno che si occupi di tutti i dettagli.

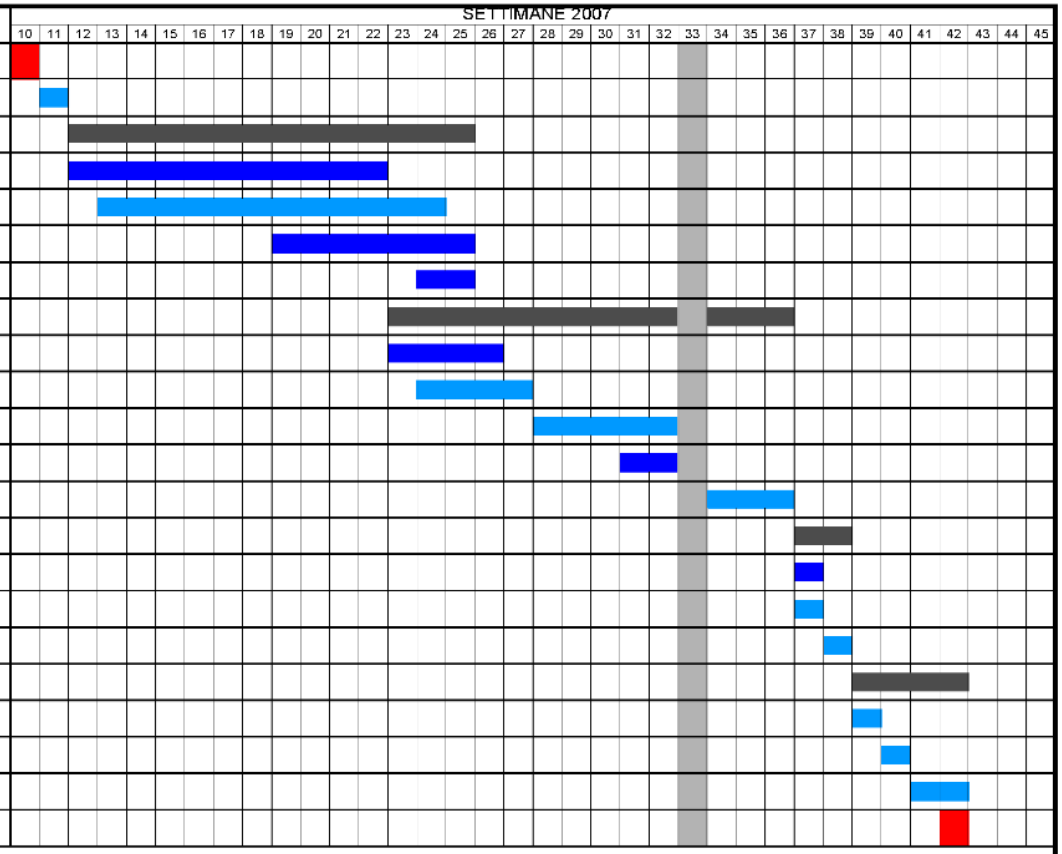
Linearità

Il successo del tuo diagramma di Gantt è strettamente legato alla mappatura lineare dell'intero progetto. Ciò significa che è necessario avere un'idea del risultato finale e dei passi da compiere prima ancora di iniziare. Può sembrare logico, ad esempio, quando si costruisce una casa, ma diventa molto più difficile se si è un team di sviluppatori web che lavora per un cliente, senza un obiettivo finale ben delineato.

Lavoro impegnativo

Come accennato in precedenza, i progetti cambiano. Se hai intenzione di utilizzare un diagramma di Gantt, dovrai adattarlo di conseguenza. Il project manager responsabile del progetto deve aggiornare frequentemente il diagramma mansione che viene semplificata utilizzando un software di gestione del progetto.

Il diagramma di Gantt usato principalmente nelle attività di project managment, è costruito partendo da un asse orizzontale - a rappresentazione dell'arco temporale totale del progetto, suddiviso in fasi incrementali (ad esempio, giorni, settimane, mesi) e da un asse verticale a rappresentazione delle mansioni o attività che costituiscono il progetto.

Delle barre orizzontali di lunghezza variabile rappresentano le sequenze, la durata e l'arco temporale di ogni singola attività del progetto. Queste barre possono sovrapporsi durante il medesimo arco temporale ad indicare la possibilità dello svolgimento *in parallelo* di alcune delle attività. Man mano che il progetto progredisce, delle barre secondarie, delle frecce o delle barre colorate possono essere aggiunte al diagramma, per indicare le attività sottostanti completate o una porzione completata di queste. Una linea verticale è utilizzata per indicare la data di riferimento. Ad ogni attività può poi essere associato un costo. Il costo può essere attribuito a una singola attività oppure si può assegnare un costo orario alle risorse, determinando il costo dell'attività in base al relativo impegno orario. Nel corso del progetto, ad ogni attività o alla risorsa può essere attribuito un costo effettivo. Dai due dati di costo, preventivato al momento della stesura del Gantt, ed effettivi, si ricavano tre curve e due indicatori di avanzamento dell'intero progetto. Le tre curve riportano la cumulata del costo preventivato e/o effettivo in funzione del tempo, ossia i costi totali effettivi e/o preventivati dall'inizio

Pos.	Descrizione attività	Inizio	Fine
1	Acquisizione ordine	2007 03 07	2007 03 07
1.1.	Apertura commessa	2007 03 07	2007 03 15
2	Acquisto materie prime	2007 03 15	2007 05
2.1.	Lamiere	2007 03 15	2007 05 28
2.1.	Forgiati	2007 03 22	2007 06 14
2.1.	Bulloneria	2007 05 07	2007 06 20
2.1.	Guarnizioni	2007 06 15	2007 06 20
3	Costruzione	2007 03 07	2007 03 15
3.1.	Calandratura lamiere	2007 05 28	2007 06 28
3.2.	Formatura fondi	2007 06 12	2007 07 04
3.3.	Saldatura	2007 07 04	2007 08 07
3.4.	Assiematura bocchelli	2007 07 30	2007 08 10
3.5.	Assiematura generale	2007 08 20	2007 09 07
4	Collaudi	2007 09 10	2007 09 19
4.1.	Esame LP	2007 09 10	2007 09 12
4.2.	Ispezione Rx	2007 09 12	2007 09 14
4.3.	Prova idraulica	2007 09 17	2007 09 19
5	Finitura	2007 09 24	2007 10 19
5.1.	Verniciatura	2007 09 24	2007 09 27
5.2.	Protezione bocchelli	2007 10 01	2007 10 05
6	Spedizione	2007 10 08	2007 10 19
7	Consegna	2007 10 19	2007 10 19