

Metriche

Ercole Colonese, 29 dicembre 2005

Nota: Questo articolo è tratta dal capitolo 17 del Manuale di sviluppo software dello stesso autore.

Le *metriche* sono adoperate per definire obiettivi misurabili e valutarne il reale raggiungimento. Questo significa che una metrica è utilizzata sia in fase iniziale per definire il “valore” di un obiettivo da raggiungere, sia a consuntivo per misurarne l’obiettivo raggiungimento (es.: la stessa metrica è utilizzata per stimare il costo del progetto in fase di pianificazione iniziale e poi, a consuntivo, per misurare il costo effettivo).

Le metriche generalmente utilizzate sono quelle riferite alle seguenti diverse tipologie:

- Metriche di prodotto;
- Metriche di progetto;
- Metriche di processo;
- Metriche di servizio.

Lo sviluppo del software, come ogni altra attività tecnica, necessita di misure e dati oggettivi per essere valutato e migliorato.

Sviluppare software senza sapere “cosa si è fatto”, “quanto si è fatto” e “come lo si è fatto” è come navigare nella nebbia senza bussola! Nessuno si metterebbe in viaggio con famiglia e bagagli senza sapere quanto dista il luogo d’arrivo, quanto carburante abbiamo in serbatoio, quanto consuma mediamente la nostra vettura e dove poter fare eventualmente rifornimento. E’ importante allora sapere “quanto” software dobbiamo sviluppare, “quale” sia la nostra produttività media, di “quante” persone avremo bisogno, “quanto” tempo occorrerà per realizzare il progetto e, in ultima analisi, “quanto” ci costerà il progetto! L’utilizzo delle metriche con metodo, rigore e coerenza è la chiave per la gestione efficace dei progetti: definire esattamente cosa dobbiamo fare, stimare quanto costa, pianificare in maniera realistica, prevedere i risultati, misurare puntualmente e agire di conseguenza. Misurare però costa, tempo e fatica. Occorre quindi resistere alla tentazione di misurare tutto e sempre (come spesso purtroppo

avviene in molte realtà, anche certificate). E' bene misurare solo ciò che serve, niente di più ma anche niente di meno. Utilizzare le metriche in maniera semplice ed efficace è anch'essa un'arte. Poche metriche, semplici ed efficaci, ma utilizzate metodicamente per costruire una base di conoscenze pregresse con cui confrontarsi. La competenza di un professionista sta nel sapere fare una stima accurata che, confrontata con i dati disponibili di progetti analoghi, dia sufficiente garanzia dei risultati.

Di seguito sono descritte le metriche più comuni disponibili. Quali di queste utilizzare di volta in volta sono riportate invece nei singoli capitoli di ciascuna fase di sviluppo (vedi voce "Metriche:").

Essenzialmente le metriche del software riguardano:

- il prodotto (codice e documentazione);
- il progetto con il quale realizzare il software;
- i processi adoperati;
- i servizi erogati (es.: manutenzione del software, assistenza agli utenti, ecc.).

Metriche di prodotto - Un prodotto software è valutato essenzialmente in base ai contenuti funzionali realizzati, alla sua capacità di funzionare correttamente e continuativamente negli ambienti previsti e al livello di aderenza agli standard stabiliti. La prima caratteristica (capacità funzionale) si riferisce al livello di implementazione dei requisiti del committente, siano essi espliciti o impliciti: "il software fa quello che ci si aspetta debba fare". La seconda caratteristica (capacità di funzionare correttamente e continuativamente negli ambienti previsti) assume diversi aspetti: il software reagisce positivamente alle variazioni dell'ambiente in cui è installato (*Affidabilità*¹), interagisce in maniera ottimale con gli utenti (*Usabilità*), utilizza in maniera ottimale le risorse necessarie (*Efficienza*), permette l'evoluzione agevole dell'applicativo (*Manutenibilità*), permette l'evoluzione tecnologica agevole (*Portabilità*), permette agli utenti di operare efficacemente (*Qualità in uso*). La terza caratteristica (aderenza agli standard) si riferisce a standard applicativi generalmente stabiliti dal Committente o dal mercato per un particolare settore d'industria (per esempio, gli standard definiti nell'ambito della Pubblica Amministrazione).

Ma un prodotto software ha anche caratteristiche intrinseche legate alla sua realizzazione: dimensioni, complessità, utilizzo di tecnologie evolute e consolidate, aderenza a standard tecnologici, ecc.

Metriche di progetto - Un progetto software ha l'obiettivo di realizzare un prodotto con caratteristiche definite, in tempi ed a costi stabiliti. Le metriche di progetto mirano quindi a misurare il grado di raggiungimento di tali obiettivi (realizzazione del prodotto richiesto, rispetto dei tempi e dei costi).

Il grado di raggiungimento del primo obiettivo (realizzazione del prodotto con le caratteristiche definite) è indirizzato dalle metriche di prodotto viste prima.

Il grado di raggiungimento del secondo obiettivo (rispetto dei tempi) è misurato tramite metriche dirette o indirette, come ad esempio: ritardo nelle consegne, correttezza delle stime, correttezza della pianificazione, efficacia del controllo dello stato di avanzamento, capacità di gestione dei rischi, adeguatezza delle risorse coinvolte, efficacia nel coordinamento delle risorse, efficacia delle comunicazioni, tempestività nelle reazioni alle situazioni critiche, ecc.

¹ Tali caratteristiche del prodotto software (Funzionalità, Affidabilità, Usabilità, Efficienza, Manutenibilità, Portabilità, Qualità in uso) sono indirizzate dallo standard ISO9126.

Metriche di processo - Un processo stabilisce come realizzare un'attività complessa come lo è, nel nostro caso, lo sviluppo di un prodotto software. Il processo definisce le fasi da seguire, le attività da svolgere, i compiti da assegnare, i ruoli da coinvolgere, i prodotti intermedi e finali da realizzare, i controlli da eseguire. Noi abbiamo aggiunto nel modello proposto i metodi e le tecniche da adottare, le metriche da utilizzare, gli strumenti da adoperare, le evidenze da produrre e conservare. L'utilizzo di un processo definito consente di prevedere i risultati intermedi e finali. Permette, quindi, di pianificare in fase iniziale le attività necessarie e di agire di conseguenza durante il suo svolgimento. La valutazione dei risultati finali ed il loro confronto con quelli attesi permette di verificare l'efficace del processo adoperato e di correggerlo e/o migliorarlo. Il processo proposto, perciò, è stabilito per essere utilizzato così com'è e quindi migliorato in base alla sua efficacia.

I processi sono misurati in termini di livello di utilizzo e risultati ottenuti. Nel processo proposto le metriche sono definite a livello di singola fase. Le misure sono effettuate manualmente o automaticamente, direttamente o indirettamente, durante lo svolgimento delle attività o a consuntivo. Alcune metriche richiedono tecniche particolari che ne aumentano l'efficacia (es. stima delle dimensioni in FP, coesione ed accoppiamento dei moduli, ecc.). L'utilizzo di strumenti automatici sono invece richiesti per effettuare misurazioni che richiederebbero un enorme dispendio di tempo se fatte manualmente (calcolo della complessità ciclomatica del software, numero di linee di codice, ecc.).

La verifica del livello di aderenza ai processi e la valutazione dell'efficacia dei processi è fatta dall'attività di Assicurazione Qualità (*Product and Process Quality Assurance*).

Metriche di servizio - Per servizio si intende, in questo modello, l'attività di manutenzione correttiva, evolutiva, adeguativa e migliorativa, nonché il supporto agli utenti. Gli indicatori che generalmente permettono di valutare la qualità del servizio sono essenzialmente legati alla prontezza con cui si accolgono le richieste e le segnalazioni di problemi, la tempestività degli interventi, l'efficacia con cui si fornisce il supporto. Per ciascun aspetto sono definite indicatori di servizio, metriche, livelli da raggiungere (soglie) e, in alcuni casi, penali da applicare in caso di mancato raggiungimento dei livelli di servizio accordati (*SLA, Service Level Agreement*).

La capacità di mantenere gli indicatori entro i limiti stabiliti dipende essenzialmente da: a) competenza delle persone e conoscenza del prodotto e dell'ambiente operativo; b) qualità e disponibilità della documentazione tecnica; c) manutenibilità del software (complessità del codice, ecc.); d) disponibilità dei casi di prova in forma organizzata (per componente/funzione/sottofunzione/ecc.).

Vediamo ora nel dettaglio le singole metriche delle diverse categorie identificate.

Metriche relative ai prodotti software

Capacità funzionale

Misura la capacità del software di “fare quello che ci si aspetta debba fare”. Una prima metrica (*completezza*) misura quindi il numero di richieste espresse (o implicitamente attese) dal committente e realizzate. Una seconda metrica misura il livello di *aderenza* a leggi, normative, standard, sicurezza, ecc. richieste.

Prima metrica: *Completezza*.

Capacità funzionale: Completezza

$$CF_C = N_{REQ-SVIL} / N_{REQ-TOT}$$

dove:

$N_{REQ-SVIL}$ = Numero di requisiti sviluppati dall'applicativo

$N_{REQ-TOT}$ = Numero totale di requisiti del committente

Nota: Il numero totale dei requisiti del committente è dedotto dal “Documento dei requisiti” come concordato con il committente stesso e tenuto aggiornato con tutti i cambiamenti apportati durante l'intero ciclo di sviluppo. Il numero di requisiti sviluppati è invece dedotto dai documenti tecnici (Specifiche funzionali, Disegno, ecc.) che descrivono quali requisiti siano stati indirizzati. Il modo corretto per gestire tale metrica è quella di creare e mantenere aggiornata la Matrice di tracciabilità dei requisiti, come previsto dal modello CMMI.

Seconda metrica: *Aderenza*.

Capacità funzionale: Aderenza

$$CF_A = N_{REQ-NORM-SVIL} / N_{REQ-NORM-TOT}$$

dove:

$N_{REQ-NORM-SVIL}$ = Numero di requisiti relativi a norme, leggi, standard, sicurezza ecc. e sviluppati dall'applicativo

$N_{REQ-NORM-TOT}$ = Numero totale di requisiti relativi a norme, leggi, ecc.

Note: Il numero totale dei requisiti relativi a norme, leggi, standard, ecc. è dedotto dal “Documento dei requisiti” se essi sono stati chiaramente e completamente individuati in fase di analisi, concordati con il committente, documentati e tenuti aggiornati durante l'intero ciclo di sviluppo. Il numero di tali requisiti sviluppati è invece dedotto dai documenti tecnici (Specifiche funzionali, Disegno, ecc.) che descrivono quali requisiti siano stati indirizzati. *Anche in questo caso, la Matrice di tracciabilità dei requisiti, mantenuta aggiornata costituisce lo strumento più adatto.*

Affidabilità (vedi pagine più sotto!!!!)

Misura la capacità del software realizzato di reagisce positivamente alle variazioni dell'ambiente in cui è installato (Affidabilità).

Usabilità

Misura la capacità del software realizzato di interagire in maniera ottimale con gli utenti (Usabilità).

Le metriche relative all'usabilità sono, forse, le meno oggettive in quanto legate alla percezione degli utenti, alle loro modalità operative personali, ecc. Fra le varie metriche predisposte, le norme ISO 9126 indicano che "... Le metriche di usabilità dovrebbero misurare gli attributi del software relative all'operatività dal punto di vista della sua facilità d'uso e dell'adattabilità agli operatori. Inoltre dovrebbero misurare la facilità di imparare il nuovo sistema e capirne il funzionamento".

Di seguito sono riportate solo alcune delle numerose metriche ISO 9126 relative all'usabilità. Una lista completa la si può ottenere direttamente dalle norme.

Caratteristica di usabilità	Metrica
Facilità di comprensione	<ul style="list-style-type: none">• Disponibilità di una Demo• Comprensione/Intuizione di quali input sono richiesti e quali output sono prodotti
Facilità di apprendimento	<ul style="list-style-type: none">• Disponibilità di un Tutorial• Facilità di imparare
Facilità d'uso (operatività)	<ul style="list-style-type: none">• Interazione attrattiva, piacevole (in USA interfacce "sexy")• Facilità di memorizzazione• Interazione guidata• Presenza di una guida per inesperti• Presenza di una guida per tutti gli utenti•• Assenza di errori• Tempo richiesto per completare un'operazione (task)• Assenza di confusione dovuta a messaggi chiari•• Capacità di recupero dopo un errore utente• Numero di tentativi per completare un'operazione (task)• Capacità di recupero

Tabella. Caratteristiche di usabilità del software (tratto da ISO9126)

Efficienza

Misura la capacità del software realizzato di utilizzare in maniera ottimale le risorse necessarie (Efficienza).

Manutenibilità

Misura la capacità del software realizzato di permettere l'evoluzione agevole dell'applicativo (Manutenibilità).

Portabilità

Misura la capacità del software realizzato di permettere l'evoluzione tecnologica agevole (Portabilità).

Qualità in uso

Misura la capacità del software realizzato di permettere agli utenti di operare efficacemente (Qualità in uso).

Dimensione di un prodotto software

Misura le dimensioni del prodotto software. Misurata da sempre in termini di migliaia di linee di codice sorgente (KLOCs – Line Of Code), da qualche decennio si adotta una nuova misura, legata al numero di funzionalità offerte dal software. Misura quindi il valore che esso rappresenta per l'utente, svincolato dalle dimensioni del codice sviluppato. Questa seconda misurazione è espressa come numero di punti funzione (Function Points – FPs). La dimensione (size) di un software è utilizzata come base per altre misure come, ad esempio, l'impegno necessario a realizzarlo, la difettosità, ecc.

Prima metrica:

Dimensione del software (1)

DIM_{KLOCs} = Numero di linee di codice, in migliaia (KLOCs)

dove: KLOCs = Kilo Lines of code

Nota: Il conteggio del numero di linee di codice di un software di certe dimensioni può essere fatto solo con l'utilizzo di appositi strumenti automatici (risulterebbe praticamente impossibile manualmente).

Seconda metrica:

Dimensione del software (2)

DIM_{FP} = Numero di punti funzione (FPs)

dove: FPs = Function Points

Nota: La tecnica per misurare il numero di punti funzione di un applicativo software è semplice nella sua formulazione di base ma può risultare complessa in caso di applicativi particolari. La tecnica è descritta nell'apposito capitolo del manuale.

Dimensione del codice sorgente

Misura le dimensioni del codice sorgente (di tutto il prodotto software oppure di quello sviluppato in una versione specifica). E' espresso in termini di numero di linee di codice, in migliaia (KLOCs) ed è utilizzato per dimensionare la produttività delle persone e da questa lo sforzo richiesto per sviluppare il prodotto.

Dimensione del codice eseguibile

Misura le dimensioni del prodotto software compilato e pronto per essere eseguito. E' utilizzato per dimensionare la memoria (su disco e/o su memoria centrale) necessario ad ospitare il prodotto software in fase di occupazione. E' misurata in numero di bytes, in migliaia o milioni (KBytes, MBytes).

Dimensione ed efficacia della documentazione

Misura le dimensioni e l'efficacia della documentazione prodotta (documentazione tecnica, manuali per l'utente, manuali per l'installazione e la gestione operativa, altra documentazione richiesta).

Una prima metrica si riferisce alle dimensioni della documentazione e misura il numero di pagine documento. E' utilizzata per valutare la produttività in fase di stesura dei manuali e per valutare lo sforzo richiesto.

Una seconda metrica si riferisce all'efficacia della documentazione che è misurata tramite il numero di scenari applicativi reali descritti, di esempi concreti riportati, di situazioni anomale da gestire, ecc.

Una terza metrica misura il livello di aderenza della documentazione prodotta agli standard definiti (utilizzo dei modelli disponibili).

Prima metrica:

Dimensione della documentazione

$$DIM_{DOC} = \text{Numero di pagine}$$

Nota: Nessuna.

Seconda metrica:

Efficacia della documentazione

$$EFF_{DOC} = N_{SCENARI} / N_{PAGG}$$

dove: $N_{SCENARI}$ = Numero totale di scenari, esempi, situazioni descritte nel manuale
 N_{PAGG} = Numero totale di pagine del manuale

Nota: Pur essendo una metrica poco oggettiva, essa risulta molto concreta in quanto praticamente di valore per gli utenti.

Metriche relative al processo di sviluppo software

Durata del progetto

Rappresenta la durata del progetto espresso in numero di mesi o, per progetti di dimensioni contenute, in settimane. E' utilizzata per valutare l'effettiva disponibilità del prodotto realizzato.

Durata del progetto

$$DUR_{PROG} = N_{MESI}$$

dove: N_{MESI} = Numero totale di mesi

Nota: La durata di un progetto è spesso misurata anche in settimane.

Produttività media

Rappresenta la produttività media delle risorse impiegate, cioè delle persone coinvolte, nelle diverse fasi del progetto. Si parla quindi di produttività media del progetto, oppure di produttività di analisi e disegno, oppure ancora di produttività di programmazione, ecc. E' misurata in termini di numero di linee di codice o di punti funzione sviluppati da una persona nell'unità di tempo stabilita (mese, settimana, giorno, ora). La produttività di una fase specifica è misurata, per esempio, in numero di linee di codice sviluppate per mese-uomo, oppure numero di casi di test eseguiti per giorno-uomo, oppure numero di pagine scritte per giorno-uomo. E' utilizzata per valutare lo sforzo richiesto per lo sviluppo del progetto a fronte delle sue dimensioni.

Prima metrica:

Produttività media (1)

$$P_{\text{MEDIA}} = \text{KLOCs} / \text{Mese_persona}$$

dove: KLOCs = Migliaia di linee di codice prodotti in un mese da una persona
Mese_persona = Unità di misura del tempo lavorativo relativo ad una persona

Nota: La produttività può essere misurata anche in giorno_persona.

Seconda metrica:

Produttività media (2)

$$P_{\text{MEDIA}} = \text{FPs} / \text{Mese_persona}$$

dove: FPs = Numero di punti funzione sviluppate in un mese da una persona
Mese_persona = Unità di misura del tempo lavorativo relativo ad una persona

Nota: La produttività può essere misurata anche in giorno_persona.

Terza metrica:

Produttività media (3)

$$P_{\text{MEDIA}} = N_{\text{PAGG}} / \text{Mese_persona}$$

dove: N_{PAGG} = Numero di pagine prodotte in un mese da una persona
Mese_persona = Unità di misura del tempo lavorativo relativo ad una persona

Nota: La produttività può essere misurata anche in giorno_persona.

Quarta metrica:

Produttività media (4)

$$P_{\text{MEDIA}} = N_{\text{TC}} / \text{Mese_persona}$$

dove: N_{TC} = Numero di casi di prova eseguiti in un mese da una persona
Mese_persona = Unità di misura del tempo lavorativo relativo ad una persona

Nota: La produttività può essere misurata anche in giorno_persona.

Sforzo richiesto per il progetto

Rappresenta il lavoro complessivo (effort) necessario per sviluppare il prodotto tenendo conto delle sue dimensioni e della produttività media delle persone coinvolte. E' quindi calcolato come rapporto tra le dimensioni del progetto e produttività media. E' utilizzato per valutare il costo, in termini di risorse, per sviluppare il progetto.

E' adoperato per calcolare anche il lavoro necessario per una singola attività: per produrre la documentazione, per eseguire un determinato numero di casi di test, ecc.

Prima metrica:

Lavoro (1): Sviluppo complessivo del prodotto

$$L_{TOT} = DIM_{PROD} / PROD_{MEDIA}$$

dove: DIM_{PROD} = Dimensione del prodotto (in FP o KLOC)
 $PROD_{MEDIA}$ = Produttività media (in FP o KLOC)

Nota: Il lavoro totale calcolato è espresso secondo l'unità di misura adottata per rappresentare la produttività (es.: una produttività espressa in mese_persona fornisce un valore totale di lavoro in mese_persona).

Seconda metrica:

Lavoro (2): Produzione della documentazione

$$L_{TOT} = DIM_{DOC} / PROD_{DOC}$$

dove: DIM_{DOC} = Dimensione della documentazione da produrre (in Numero di pagine)
 $PROD_{DOC}$ = Produttività media (in Numero di pagine per Mese_persona)

Nota: Il lavoro totale calcolato è espresso secondo l'unità di misura adottata per rappresentare la produttività (es.: una produttività espressa in mese_persona fornisce un valore totale di lavoro in mese_persona).

Terza metrica: Analoga per l'esecuzione dei casi di test.

Numero di cambiamenti apportati

Rappresenta il numero di modifiche apportate al progetto in corso d'opera. Le modifiche si riferiscono a elementi già consolidati e possono riguardare i requisiti, le funzionalità, il disegno, i manuali. Sono generati dalla necessità di aggiungere un requisito nuovo oppure di modificare o cancellarne uno esistente. La necessità può derivare da una richiesta esplicita del committente oppure da una errata o incompleta interpretazione del fornitore. Un requisito nuovo (o modificato o cancellato) provoca, quasi sicuramente, una o più modifiche alle funzionalità, al disegno, al codice ed alla documentazione per l'utente. E' importante, quindi, misurare il numero di modifiche apportate in corso d'opera per valutare gli impatti sui tempi di realizzazione e sui costi del progetto.

Modifiche = Numero di modifiche ai requisiti (o funzionalità, o disegno, o codice, o manuali) / Numero totale di requisiti (o funzionalità, o pagine di manuale).

Nota: per le modifiche al disegno o al codice il numero di modifiche è un valore assoluto.

Numero di errori rilevati

Rappresenta il numero di errori rilevati nel prodotto durante le diverse fasi di sviluppo. Così si parla di numero di errori rilevati nel disegno, nel codice, nella documentazione, oppure nei casi di test, ecc. Gli errori sono rilevati sulla documentazione si riferiscono alle attività di revisione e ispezione tecnica, quelli di codice alle attività di test. Sono importanti in quanto permettono di calcolare l'efficacia del processo di revisione e di test e, da questa, la curva di rimozione degli errori durante lo sviluppo. Dalla curva di rimozione degli errori è possibile prevedere, con tecniche più o meno sofisticate, il tasso di errori residuo, ovvero il numero di errori che gli utenti troveranno in fase di esercizio del prodotto. Il numero di errori rilevati è normalizzato rispetto alle dimensioni del progetto.

$$\text{Numero di errori} = \text{Numero di errori rilevati} / \text{Dimensioni del progetto (esprese in KLOCs o FPs)}$$

Copertura dei test

Rappresenta il livello di copertura che i test eseguiti forniscono rispetto alle funzionalità offerte dal prodotto. E' misurato come rapporto tra le funzionalità effettivamente verificate dai casi di test eseguiti ed il numero totale delle funzioni disponibili nel prodotto ed è espresso in punti percentuale. Una buona copertura è pari al 100%. Può essere calcolata anche come rapporto tra il numero totale di casi di test eseguiti ed il numero totale delle funzionalità offerte dal prodotto. In questo caso il valore può superare il 100%. Questa seconda misurazione evidenzia che alcune funzioni sono testate con più di un caso di test, e deve essere sempre accompagnata dalla prima (copertura) che dimostra quante funzioni sono state testate.

$\text{Copertura del test (1)} = \frac{\text{Numero di funzioni testate}}{\text{Numero totale di funzioni disponibili}}$
$\text{Copertura del test (2)} = \frac{\text{Numero di casi di test eseguiti}}{\text{Numero totale di funzioni disponibili}}$

Efficacia dei test

Rappresenta la capacità di rilevare errori nel prodotto tramite le attività di test. E' misurata come rapporto tra il numero di errori rilevati ed il numero di casi di test eseguiti. Essa è comunque una misurazione qualitativa e non quantitativa. Infatti, il numero di errori rilevati tiene conto solo degli errori effettivamente riconosciuti come tali e di cui si fornisce una correzione del codice. Sono perciò esclusi dal conteggio gli errori duplicati di altri errori, quelli dovuti ad una operazione sbagliata del testatore, quelli dovuti ad una errata impostazione del sistema (ambiente, base dati, ecc.), ecc. Per riferirsi a questa misurazione si usa il termine di "errori validi". In ambienti di test più evoluti si calcola anche il rapporto tra il numero di errori "validi" ed il numero totale di errori rilevati. Un'altra misurazione dell'efficacia del test è rappresentata dal rapporto tra il numero di errori validi rilevati in fase di test e le dimensioni del prodotto.

$\text{Efficacia del test (1)} = \text{Numero di errori validi} / \text{Numero di casi di test eseguiti}$
$\text{Efficacia del test (2)} = \text{Numero di errori validi} / \text{Numero totale di errori rilevati}$
$\text{Efficacia del test (3)} = \text{Numero di errori validi} / \text{Dimensioni del prodotto}$

Efficacia delle revisioni

Rappresenta la capacità di rilevare errori nell'analisi e nel disegno del prodotto tramite le attività di revisione ed ispezioni tecniche. E' misurata come rapporto tra il numero di errori rilevati durante la revisione e le dimensioni del prodotto. Ovviamente le dimensioni del prodotto in fase di analisi e disegno rappresenta la stima iniziale, oppure di pagine revisionate.

$\text{Efficacia delle revisioni (1)} = \text{Numero di errori} / \text{Dimensioni del prodotto}$
$\text{Efficacia delle revisioni (2)} = \text{Numero di errori} / \text{Numero di pagine ispezionate}$

Metriche relative al processo di manutenzione del software

Severità dei problemi

Rappresenta la gravità assegnata ad un difetto rilevato in esercizio in relazione al grado di deterioramento delle funzionalità dell'applicazione. Generalmente si utilizzano tre livelli di severità. Ovviamente si richiedono tempi di risoluzione diversi a seconda della severità (gravità) del problema. Ecco le definizioni più comuni.

Severità	Descrizione
1	L'applicazione software non è più disponibile agli utenti (nessuna funzione può essere eseguibili in alcun modo) causando seri impatti sull'operatività degli utenti.
2	Le funzioni applicative non sono fortemente limitate causando impatti significativi sull'operatività degli utenti.
3	Alcune funzionalità rilevano errori nel funzionamento causando impatti minori sull'operatività degli utenti.

Tempo di presa in carico dei problemi

Rappresenta la capacità di reazione da parte del gruppo di manutenzione alla rilevazione dei difetti rilevati durante l'utilizzo del software, da parte degli utenti, una volta messo in esercizio. E' misurata come intervallo di tempo tra la notifica del difetto e la sua presa in carico. Il tempo è rilevato generalmente da un tool per la registrazione dello stato delle chiamate. I tempi sono registrati in "data-ora-minuti".

$$\text{Tempo di presa in carico} = T_1 - T_0$$

(dove T_1 = tempo di presa in carico; T_0 = tempo di notifica)

Tempo di risoluzione dei problemi

Rappresenta la capacità di risolvere prontamente i difetti rilevati durante l'utilizzo del software, da parte degli utenti, una volta messo in esercizio e presi in carico da parte del gruppo di manutenzione. E' misurata come intervallo di tempo tra la presa in carico del problema e la sua risoluzione. Il tempo è rilevato generalmente da un tool per la registrazione dello stato delle chiamate. I tempi sono registrati in "data-ora-minuti". I tempi di risoluzione dipendono dalla severità dei problemi.

$$\text{Tempo di risoluzione (Severità } x) = T_2 - T_1$$

(dove T_2 = tempo di risoluzione; T_1 = tempo di presa in carico)

Rispetto dei tempi di consegna

Rappresenta la capacità di rispettare i tempi di consegna concordati per gli interventi di manutenzione evolutiva e sviluppo. E' misurata come differenza dei tempi di consegna effettiva e pianficata.

$$\text{Rispetto dei tempi di consegna} = D_1 - D_0$$

(dove D_1 = data di consegna effettiva; D_0 = Data di consegna prevista)

Metriche relative alla qualità del software

Tali metriche permettono di valutare le qualità intrinseche del software sviluppato (es.: complessità, aderenza a standard etc.) e la corrispondenza ai requisiti dichiarati. In particolare, le metriche più significative sono:

- complessità ciclomatica
- livello di accoppiamento
- livello di coesione
- difettosità residua
- usabilità
- efficienza
- robustezza
- recuperabilità
- manutenibilità

Vediamone i singoli significati.

Complessità ciclomatica

Rappresenta il livello di complessità di un modulo (programma) calcolato rispetto ad un modello strutturato, cioè secondo le regole della programmazione strutturata che tutti i programmatori conoscono. E' calcolata sul grafo che rappresenta la logica interna del modulo dove si prendono in considerazione i cammini logici percorsi dal software all'interno del modulo ed i nodi presenti, cioè i punti decisionali del programma. Il numero ciclomatico è stato introdotto da Tom McCabe e perfezionato da altri. Dal numero ciclomatico puro è stata derivata una seconda misurazione pratica che è quella effettivamente usata: la complessità ciclomatica "essenziale" di un modulo che è la complessità ciclomatica base calcolata sul grafo dopo aver eliminato tutti i cammini strutturati. Mentre per la complessità ciclomatica base Tom McCabe raccomandava valori inferiori a 10, per la complessità ciclomatica essenziale oggi si raccomandano mediamente valori non superiori a 4.

$$\text{Complessità ciclomatica di un modulo } v(G) = e - n + 2$$

(dove e = cammino, n = nodo)

Livello di accoppiamento

Rappresenta il grado di conoscenza che un modulo/programma ha su di un altro modulo/programma chiamante o chiamato. Esso definisce, cioè, se la logica di un programma dipende dalla logica di un altro programma. L'accoppiamento è bene che assuma, quindi, un valore basso. Esiste una scala di valori di accoppiamento a 6 livelli definiti per:

1. **dato:** La comunicazione tra i moduli avviene esclusivamente tramite parametri e codici di ritorno (esempio: *CALL PGM (P1, P2, P3, RC)*).
2. **strutture dati:** La comunicazione tra i moduli avviene tramite l'accesso condiviso ad una struttura dati globale (esempio: *Logical Common Area*).
3. **controllo:** La comunicazione tra i moduli avviene tramite parametri di esecuzione (esempio: *Switch*).
4. **elementi esterni:** Il modulo fa riferimento ad un simbolo definito in un altro modulo, dichiarandolo, per esempio: *External*.
5. **aree dati comuni:** Il modulo condivide con altri moduli l'accesso ad una intera area dati (esempio: *Data Common Area*).
6. **contenuto:** Il modulo fa riferimento diretto alla struttura fisica di un altro modulo (esempio: salto ad un indirizzo - *label* - di un altro programma).

Livello di coesione

Rappresenta il grado di dipendenza funzionale tra le parti che compongono un modulo/programma, cioè il grado di attinenza delle sue varie parti. Occorre, quindi, che i programmi abbiano un alto valore di coesione. Esistono 7 livelli di coesione, in ordine crescente:

1. **Casuale:** Nel modulo esistono parti distinte di codice non correlate tra di loro e senza motivi validi che lo giustifichino (esempio: funzioni diverse);
2. **Associazione logica:** Il modulo contiene del codice che realizza funzioni distinte ma correlate (esempio: aggiornamento di tabelle correlate ma di funzioni diverse);
3. **Temporale:** Il modulo contiene del codice logicamente distinto, ma da eseguire nello stesso momento (esempio: inizializzazioni);
4. **Procedurale:** Il modulo esegue funzioni correlate dal punto di vista dell'utente;
5. **Comunicazione:** Nel modulo esiste una dipendenza funzionale tra le parti che lo compongono;
6. **Sequenziale:** Il modulo esegue più funzioni distinte che vanno eseguite in stretta sequenza;
7. **Funzionale:** Il modulo esegue una ed una sola funzione (macro-funzione).

Difettosità del software sviluppato

Rappresenta il numero di errori residui che si suppone abbia il software una volta rilasciato in esercizio. Il tasso di difettosità residuo è ovviamente una stima e potrà essere realmente misurata solo a consuntivo contando il numero di anomalie rilevate dagli utenti in un periodo di tempo stabilito (per esempio nei primi sei mesi di utilizzo oppure nel primo anno di esercizio). Al momento del rilascio del prodotto la stima degli errori residui può essere fatta con tecniche diverse più o meno sofisticate che qui non descriviamo. Il numero a consuntivo, invece, è normalizzato con le dimensioni del prodotto (numero di KLOCs o di FPs).

$$\text{Difettosità del software sviluppato} = N / S$$

(dove N = numero di difetti rilevati; S = dimensioni del software sviluppato espresso in FP o Kloc)

Nota: se il periodo di garanzia è lungo (per esempio un anno) si calcola la difettosità in sottoperiodi (esempio, trimestri). La tabella che segue mostra, per esempio, valori soglia della difettosità nei trimestri.

Difettosità	1° trim.	2° trim.	3° trim.	4° trim.
Valore soglia (num. Errori/MLoc)	5	3	1	0

Usabilità

Rappresenta la facilità di utilizzo del prodotto. E' un giudizio soggettivo espresso da uno o più utenti sulla propria percezione della facilità d'uso delle funzionalità sperimentate. Per ridurre la soggettività della misura si calcola la media del giudizio espresso da almeno cinque utenti diversi. Il giudizio è espresso utilizzando una scala di valori predefinita (per esempio, su una scala a tre valori: 1 = poco usabile, 2 = mediamente usabile, 3 = molto usabile, oppure 1= bassa usabilità, 2 = media usabilità, 3= alta usabilità, ecc.). Il giudizio espresso dagli utenti coinvolti dipende da fattori personali (es.: cultura, coinvolgimento, conoscenza, ecc.) e dalle attività svolte per esprimere il giudizio (es.: test di usabilità, oppure revisione di un prototipo). Si raccomanda, quindi di scegliere gli utenti da coinvolgere in numero e rappresentatività adeguata e di eseguire i test simulando quanto più possibile una situazione reale di utilizzo del prodotto (es.: caso d'uso reale e concreto).

Usabilità = Valore medio della valutazione espressa dagli utenti coinvolti su una scala predefinita di valori di usabilità (per esempio su una scala a tre valori di usabilità: bassa, media, alta)

Efficienza

Rappresenta le prestazioni del prodotto, cioè la sua capacità di reazione alle richieste dell'utente. Sono conosciute più propriamente come "performance" del prodotto. E' una misura oggettiva e si riferisce, generalmente ai tempi di risposta di una transazione oppure all'utilizzo di risorse di sistema da parte dell'applicazione per eseguire le funzionalità richieste.

<i>Performance (1)</i> = Tempo medio di risposta di una transazione (in secondi)
<i>Performance (2)</i> = Numero di spazio richiesto su disco o su memoria centrale
<i>Performance (3)</i> = % di utilizzo delle linee di trasmissione impiegate

Robustezza

Rappresenta la capacità del prodotto di continuare a funzionare senza degrado nelle sue prestazioni anche in condizioni particolari (esempio: uso continuativo 24 ore su 24, 7 giorni su 7), oppure in condizioni limite (esempio: 500 utenti collegati contemporaneamente). Si misurano le prestazioni del prodotto (tempi di risposta ed utilizzo delle risorse) e si verifica che non ci sia degrado nelle prestazioni. Valgono le stesse metriche definite per calcolare le prestazioni.

<i>Performance (reattività)</i> = Tempo medio di risposta di una transazione (in secondi)
<i>Performance (utilizzo memoria)</i> = Numero di spazio richiesto su disco o su memoria centrale
<i>Performance (utilizzo linee)</i> = % di utilizzo delle linee di trasmissione impiegate

Recuperabilità

Rappresenta la capacità del prodotto di recuperare una condizione anomala quando questa si presenti (esempio: ripristino dei dati di partenza nel caso di caduta durante l'esecuzione di una transazione). Le capacità di recupero possono essere automatiche, semi-automatiche, manuali, assenti. Ovviamente un prodotto senza capacità di recupero delle situazioni anomale è di bassa qualità e potrebbe non essere accettato. Le metriche sono quindi:

<i>Recuperabilità = Automaticamente, Semiautomatica, Manuale, Assente</i>

Manutenibilità

Rappresenta la facilità di eseguire la manutenzione (correttiva o evolutiva) da parte di personale qualificato ma non necessariamente esperto del prodotto specifico. Nel caso di manutenzione correttiva (quella che interessa questo processo specifico) la manutenibilità può essere misurata tramite i seguenti parametri:

Tempo medio di risoluzione delle anomalie in base alla loro gravità;

Numero di correzioni di anomalie non andate a buon fine (che richiedono, cioè, un secondo intervento correttivo) rispetto al numero totale di anomalie risolte

$Manutenibilità (1) = \text{Tempo risoluzione medio} / \text{Totale anomalie risolte}$
$Manutenibilità (2) = \text{Anomalie risolte con riciclo} / \text{Totale anomalie risolte}$

Metriche object-oriented

Rappresentano i diversi aspetti che caratterizzano il software sviluppato con la tecnologia ad oggetti. Fra quelle elencate qui di seguito le metriche maggiormente utilizzate sono le prime cinque:

AV(g)	Media della Complessità Ciclomatica
CBO	Coupling Between Object classes
NOC	Number of Children
LOC	Lines of Code
LOCM	Lack of Cohesion of Methods
DEPTH	Depth
WMC	Weighted Methods Per Class
LCOM	Lines of COMment
RFC	Response For a Class

Proposta di metriche OO e soglie rilevabili con McCabe

Il tool McCabe IQ supporta la raccolta di un elevato numero di metriche relative a software sviluppato con linguaggi OO. In particolare:

INCAPSULAMENTO

1. Percent Public Data (PCTPUB) - PCTPUB rappresenta la percentuale di dati di tipo PUBLIC e PROTECTED di una classe;
2. Access to Public Data (PUBDATA) - PUBDATA indica il numero di accessi ai dati di tipo PUBLIC e PROTECTED;

POLIMORFISMO

3. Percent of Unoverloaded Calls (PCTCALL) - PCTCALL è il numero di chiamate non-overloaded del sistema;
4. Number of Roots (ROOTCNT) - ROOTCNT è il numero totale di radici di gerarchia della classe nell'ambito di un programma;
5. Fan-in (FANIN) - FANIN è il numero di classi da cui una classe è derivata.

QUALITA'

6. Average V(g) (AV(g)) – AV(g) è la media dei valori della Complessità Ciclomatica dei metodi di una classe;
7. Lines Of Code (LOC) – LOC è la media delle linee di codice eseguibile che implementano i metodi di una classe

8. Lines of COMment (LCOM) – LCOM è la media delle linee di commento nei metodi di una classe, rispetto alle linee di codice eseguibile;
9. Average ev(G) (AEV) - AEV è la media dei valori della Complessità Essenziale dei metodi di una classe;
10. Hierarchy Quality (QUAL) - QUAL è il conteggio del numero di classi di un sistema che risultano dipendenti da propri “discendenti”.
11. Weighted Methods Per Class (WMC) - WMC è il numero di metodi implementati da una classe;
12. Depth (DEPTH) - DEPTH indica a quale livello una classe è posizionata in una gerarchia di classi;
13. Number of Children (NOC) - NOC è il numero di classi che sono derivate direttamente da una classe specificata;
14. Coupling Between Object classes (CBO) – è una misura di come i metodi di una classe interagisce con i metodi di un'altra classe;
15. Response For a Class (RFC) - RFC è un conteggio dei metodi effettuati all'interno di una classe più il numero di metodi accessibili ad un oggetto di questo tipo di classe dovuta all'ereditarietà.
16. Lack of Cohesion of Methods (LOCM) - LOCM è la misura di come i metodi interagiscono con i dati in una classe.

Non tutte queste metriche vantano un livello di utilizzo sufficientemente esteso da rendere disponibili criteri di valutazione delle misurazioni sufficientemente sperimentati ed affidabili. E' per questa ragione che il RTI utilizzerà soltanto un sottoinsieme di tali metriche per misurare la qualità del software prodotto, determinarne l'accettabilità e governare l'esecuzione delle procedure di assicurazione qualità. Per ciascuna metrica utilizzata viene definito un “Valore di accettabilità” che determina l'accettabilità o meno dell'oggetto o dell'insieme degli oggetti misurati (nel caso si tratti di una media). Per alcune metriche è definito anche un “Valore di attenzione” che, pur non comportando la non accettabilità dell'oggetto o dell'insieme degli oggetti misurati, determina l'esecuzione di attività di indagine specifiche per identificare l'esistenza di rischi o problemi potenziali.

Metrica	Descrizione	Valore di accettabilità	Valore di attenzione
AV(g)	Media della Complessità Ciclomantica dei metodi della classe	≤ 6	$5,5 \leq AV(g) \leq 6$
LOC	Lines of Code	≤ 30	$28 \leq LOC \leq 30$
LCOM	Lines of COMment	$\geq 30\%$	$30 \leq LCOM \leq 32$
WMC	Weighted Methods Per Class	≤ 14	$12 \leq WMC \leq 14$
RFC	Response For a Class	≤ 100	$90 \leq RFC \leq 100$
LOCM	Lack of Cohesion of Methods	≥ 75	$75 \leq LOCM \leq 80$
CBO	Coupling Between Object classes	≤ 2	$= 2$
DEPTH	Depth	≤ 7	$6 \leq DEPTH \leq 7$
NOC	Number of Children	≤ 3	$= 3$