Ercole Colonese

Verifica e validazione del software negli standard e modelli di maturità

Una panoramica di come sia trattato il tema del testing del software nei diversi standard internazionali e nei modelli di maturità

Versione 1.0 – Giugno 2006

4.1

Note sulla versione 1.0

La presente versione (1.0) descrive come il tema del testing del software sia trattato negli standard internazionali e nei vari modelli di maturità esistenti (almeno in quelli conosciuti dall'autore).

Ringraziamenti

Ringrazio poi in anticipo tutti coloro che vorranno inviare segnalazioni, commenti e suggerimenti, per correggere e migliorare il manuale.

Versioni del manuale

2006	2007	2008	2009	2010
v1.0				

Manuali di sviluppo software

Il presente manuale fa parte di una serie di documenti, in parte già disponibili ed in parte in fase di realizzazione, che mirano a fornire una vista dei vari temi dello sviluppo software. Essi riportano l'esperienza pratica maturata dall'autore in molti anni di lavoro. L'elenco dei manuali e del relativo stato di completamento riportato qui di seguito si riferisce alla data in cui è rilasciata la presente versione del manuale in oggetto.

1. Sviluppare software oggi
2. Strategia di sviluppo e ciclo di vita del software
3. Gestione dei progetti software
4. **Qualità, test e collaudo del software**
5. Test e collaudo del software in Rete (e-Testing)
6. Qualità del software
7. Modello ISO 9126 per la qualità del software
8. Metriche del software
9. Usabilità del software
10. Usabilità dei siti Web
11. Metodi e tecniche dello sviluppo software
12. Introduzione al modello CMMI
13. Glossario dei termini dello sviluppo software
14. Bibliografia

I manuali completati sono disponibili nel sito dell'autore:
(http://www.colonese.it/pubblicazioni/htm).

A chi sono rivolti i manuali

I manuali sono rivolti a:

- **Studenti universitari** che seguono i corsi di ingegneria del software e che necessitino di approfondire i temi della qualità del software e del test e collaudo in particolare;
- **Responsabili IT dei clienti** che necessitino di una guida di riferimento per le attività di loro pertinenza nell'ambito dei progetti software. In particolare, i dirigenti ed i tecnici che necessitino di approfondire i temi relativi alla loro collaborazione con il fornitore;
- **Capi progetto** (**Project Manager**) che necessitino di una guida di riferimento per le attività di pianificazione e controllo delle attività di sviluppo del software in generale e del test e collaudo del software in particolare;
- **Sviluppatori**, **collaudatori**, **personale dell'assicurazione qualità** e **consulenti di processo** che necessitino di approfondire i temi relativi al loro ruolo.

4 Software testing

Sommario

Indice degli argomenti

Indice delle figure

Indice delle tabelle

Non è presente alcuna tabella.

1. Introduzione

1.1 A chi è rivolto il documento

Questo articolo è una parte di un manuale più completo che tratta in maniera esaustiva il tema del testing del software.

Esso, come tutti gli altri manuali, è rivolto a studenti e professionisti.

I **tester**, primi fra tutti, possono trovare gli elementi necessari per approfondire la materia di loro responsabilità: progettare i test, utilizzare tecniche e metodi per migliorare i test, adoperare metriche del software e del testing per valutare l'efficacia del lavoro, ecc. I tester possono porre i contenuti del manuale come base per la crescita professionale nel loro ruolo.

I **capi progetto** (**Project Manager**) possono trovare nel manuale una guida di riferimento per le attività di pianificazione e controllo delle attività relative alla qualità del software, il test e collaudo. Inoltre, possono trovare gli spunti necessari per capire l'importanza della loro collaborazione con il cliente e gli elementi critici per il successo del progetto.

I **responsabili IT dei clienti**, a loro volta, possono trovare nel manuale una guida di riferimento per le attività di loro pertinenza nell'ambito dei progetti software. In particolare, il manuale pone in rilevo l'importanza della collaborazione tra cliente e fornitore per il successo del progetto. A tale fine, i dirigenti ed i tecnici possono approfondire i temi relativi alle loro responsabilità e trovare i dettagli necessari.

Gli **sviluppatori** coinvolti nelle attività di test possono approfondire i temi relativi al loro ruolo con particolar riferimento ai concetti di qualità in generale e

di test e collaudo in particolare. Un buon programmatore dovrebbe conoscere molto bene anche il tema della qualità del software e quello del test e collaudo. Entrambi sono complementari all'attività di programmazione. Quindi, completano e arricchiscono la sua professionalità.

Gli **studenti universitari** che seguono i corsi di ingegneria del software e che necessitino di approfondire i temi della qualità del software, del test e collaudo in particolare possono trovare spunti per lo studio e la ricerca.

1.2 Verifica e validazione nel processo di sviluppo del software

Il processo di sviluppo software è generalmente definito all'interno del sistema di gestione per la qualità (SGQ) aziendale. Lo standard ISO 9001:2000 include:

- i processi produttivi (processi di sviluppo del software),
- i processi di verifica e validazione (ispezioni, test e collaudo) e
- i processi di assicurazione qualità (Audit del SGQ).

L'assicurazione qualità è svolta nell'ambito delle verifiche ispettive interne (V.I.I.). La valutazione di quanto i progetti siano aderenti agli standard definiti dal sistema di qualità aziendale permette di correggere le eventuali deviazioni dai processi stabiliti e di migliorarli, anche a fronte dei risultati ottenuti in termini di qualità del prodotto finale.

Standard di documentazione

Gli standard relativi alla documentazione sono molto importanti in un progetto software. Il software, infatti, per la sua peculiarità, è un'entità astratta che si concretizza man mano che si procede nel ciclo di vita. Le idee iniziali del cliente (necessità, problemi, opportunità, ecc.) si traducono in un documento di requisiti. Questo, a sua volta, si traduce in una soluzione descritta in altri documenti (specifiche funzionali e disegno). Per finire, la soluzione prende forma nel codice sorgente, anch'esso assimilabile ad un documento. Solo al momento dell'esecuzione del codice sviluppato si "vede" il risultato in termini di funzionalità offerte e comportamento del sistema.

La traduzione di idee (requisiti) in un oggetto concreto (software) è un'attività complessa e di estrema criticità. Si tratta di un processo i cui elementi sono tutti aleatori: comunicazione delle esigenze, interpretazione di quanto ascoltato, traduzione in linguaggio naturale (o formale) e traduzione successiva in linguaggi macchina interpretabili dal sistema. Niente di più aleatorio e soggettivo. L'utilizzo di modelli per la produzione della documentazione diventa quindi importantissimo per garantire strutturazione, completezza e correttezza del nostro lavoro di "traduttori di esigenze".

Per standard di documentazione intendiamo i modelli per la loro scrittura e le procedure per l'identificazione, la revisione, l'approvazione, la diffusione, la conservazione e l'aggiornamento.

Standard di verifica e validazione

La "verifica dei documenti" tramite revisioni tecniche - che siano svolte in maniera formale o informale poco importa purché siano svolte correttamente - rappresenta l'unica azione in nostro possesso per garantire la correttezza del nostro operato.

L'efficacia delle verifiche si valuta in base al numero di errori scoperti con le revisioni tecniche dei documenti ispezionati. I valori sono generalmente dedotti dall'esperienza e dall'uso di "profili di qualità del prodotto". Tutte le ispezioni pianificate devono essere svolte e tutti gli errori scoperti devono essere corretti perché l'attività di verifica possa essere considerata completata con successo.

La "validazione del software" tramite i test e collaudi rappresenta infine lo strumento essenziale per valutare concretamente la qualità raggiunta dal prodotto sviluppato.

L'efficacia dei test si verifica tramite il livello di copertura che i casi di test progettati sono in grado di garantire rispetto ai requisiti ed alle specifiche (vedi "Matrice di test"). Un secondo parametro che aiuta a verificare l'efficacia dei test è il numero di errori scoperti e risolti durante la loro esecuzione. I valori sono generalmente dedotti dall'esperienza e dall'uso di "profili di qualità del prodotto". I criteri di uscita delle fasi di test richiedono che tutti i casi di test pianificati siano stati completati con successo e che tutti gli errori scoperti siano stati corretti e validati.

Standard di auditing

Compito dell'assicurazione qualità è quindi controllare che tutto quanto abbiamo detto fin'ora sia correttamente svolto da tutti i progetti. Se la qualità finale prodotta con tali processi è quella attesa vuol dire che i nostri processi sono buoni. Altrimenti occorrerà migliorare i processi e innescare un altro giro di controlli e verifiche. Fino a raggiungere il livello di qualità atteso. Condizione necessaria e imprescindibile di tale approccio è che tutti i progetti adottino gli standard stabiliti. In caso contrario non sarà possibile discernere tra risultato negativo prodotto da processi insufficienti e scarsa qualità dovuta a mancato utilizzo dei processi stabiliti.

Conclusione: l'assicurazione qualità è lo strumento che ci permette di controllare, passo passo, che la qualità finale sarà quella attesa. E di intervenire per tempo in caso di deviazione!

1.3 La qualità del software in pratica

1.3.1 La qualità nei progetti quotidiani

La qualità del software ha due aspetti, uno esterno ed uno interno al prodotto cui si riferisce.

L'aspetto "esteriore" del software è quello che gli utenti sperimentano quando utilizzano il prodotto e fa riferimento alle funzionalità, all'affidabilità, all'usabilità e all'efficienza. La *funzionalità*, a sua volta, è valutata in base alle esigenze ma anche alle aspettative degli utenti. Tra queste sono considerate la completezza e la ricchezza delle funzioni rispetto alle necessità, ai requisiti espressi; l'accuratezza con la quale le funzioni sono rese disponibili; la possibilità di utilizzare le funzionalità in maniera integrata con altre applicazioni presenti sul sistema e con le quali l'utente deve interagire nel completare i propri compiti di lavoro. *L'affidabilità* è un altro aspetto esteriore del software ed è valutato dagli utenti in base alla capacità dell'applicazione di funzionare correttamente - senza cadute del sistema o, come suol dirsi in gergo, "senza che vada in crash" - per tutto il tempo in cui l'applicazione serve, cioè per tutto il tempo di lavoro. In caso di caduta, poi, è richiesto che all'atto della ripartenza del sistema non ci sia alcuna perdita di dati e che gli utenti posano ripartire esattamente dal punto in cui erano arrivati nel lavoro, recuperando le transazioni eseguire prima della caduta. L'*usabilità* è un'altra caratteristica esterna del software di grande importanza, valutata dagli utenti in maniera soggettiva ma non per questo in maniera meno critica. Di essa gli utenti valutano la facilità con cui sono in grado di capire e intuire le nuove funzionalità disponibili, ad imparare e a ricordarle, ad adoperale con successo. L'*efficienza*, infine, ha un impatto sugli utenti in termini di tempi di risposta del sistema. Al riguardo siamo tutti consapevoli dell'impatto negativo che tempi troppi lunghi hanno sulla concentrazione che gli utenti riescono a mantenere sull'attività in corso e sulla produttività. L'impiego di risorse del sistema (memoria, spazio disco, rete), da parte dell'applicazione, oltre quelle previste è anch'esso valutato negativamente.

La qualità "intrinseca" del software, invece, è legata alle sue caratteristiche interne e sono visibili sono a chi deve operare su di esso - l'architettura, i componenti ed il codice -. E' l'attività principale svolta dai programmatori in fase di manutenzione, sia correttiva che evolutiva, perfettiva o migliorativa. In questa fase i programmatori devono "entrare nel codice" e modificarlo. Il loro compito è facilitato o reso difficile - "impossibile!" - proprio da quelle caratteristiche del software dette manutenibilità e portabilità. La *manutenibilità* valuta la facilità con cui i programmatori riescono a diagnosticare i problemi e ad intervenire sul codice per correggere gli errori – "debugging" -, la stabilità del software a rimanere funzionante anche dopo le modifiche, la facilità ad eseguire test delle modifiche effettuate coinvolgendo il meno possibile il resto del codice. L'ingegneria del software definisce caratteristiche di qualità specifiche a tale riguardo come, ad esempio, la leggibilità del codice, la sua la complessità, la stan-

dardizzazione, la presenza di commenti, ecc. La *portabilità*, invece, è legata alla facilità con cui il software può essere portato su piattaforme tecnologiche diverse da quelle originali per seguire l'evoluzione tecnologica del sistema. Le caratteristiche specifiche che i programmatori apprezzano in questa fase sono la capacità del software di "adattarsi" al nuovo sistema tecnologico, la facilità di installazione nel nuovo ambiente, la conformità alle regole e standard del nuovo sistema, ecc.

Le norme ISO(IEC 9126 definiscono il modello della qualità del software specificando le caratteristiche interne, esterne ed in uso. Il modello è descritto in un apposito paragrafo del capitolo successivo di questo manuale.

Le caratteristiche di qualità del software sopra elencate, esplicitamente espresse o implicitamente richieste, devono essere opportunamente indirizzate dal progetto di sviluppo e diventare oggetto di analisi, progettazione, codifica e test del software. In particolare, le suddette caratteristiche di qualità devono essere opportunamente descritte in termini di requisiti ed essere tradotte in obiettivi misurabili con tanto di valore target da raggiungere. Solo così siamo certi di sviluppare un software che possieda le caratteristiche di qualità attese dal cliente e ritenute importanti ai fini dell'accettazione finale del prodotto.

La qualità è quindi realizzata e controllata lungo l'intera durata del progetto e non sarà più il risultato "casuale" da scoprire, con apprensione, alla fine del progetto!

1.3.2 La qualità valutata dal cliente

Se la qualità è il risultato del lavoro finale, al termine di un progetto ci si deve attendere dai nostri clienti giudizi corrispondenti al livello di qualità prodotto. La valutazione finale è espressa con un giudizio più o meno positivo a seconda del grado con cui abbiamo soddisfatto le aspettative. Di seguito sono riportate frasi che indicano il miglior giudizio che ci si potrebbe attendere. Nella maggior parte dei casi – purtroppo – i nostri progetti non raggiungono tutti gli obiettivi ed i giudizi del cliente saranno diversi da quelli espressi qui di seguito.

(1) "La soluzione realizzata indirizza pienamente e correttamente i bisogni espressi dal nostro business".

(2) "Il prodotto finale incontra le aspettative dei nostri utenti che si sono dichiarati pienamente soddisfatti, sia della completezza delle funzionalità sviluppate che della semplicità e facilità nel loro utilizzo".

(3) "La soluzione sviluppata si integra perfettamente con le applicazioni esistenti; garantisce un alto livello di sicurezza, come richiesto dal nostro business; ha buone prestazioni senza consumare risorse eccessive; risulta essere robusta ed affidabile quanto basti per garantire la produttività delle nostre persone evitando attività ripetitive, inutili o dannose; è flessibile e ci permette di espanderci in futuro secondo le esigenze di business che il mercato ci richiederà".

Fermiamoci qui e vediamo come tali giudizi – con un processo a ritroso - possono essere tradotti in requisiti ed obiettivi di progetto. Qui si fa riferimento alle norme ISO 9126 che definisce il modello della qualità del software, le categorie e le caratteristiche.

1.4 Standard e modelli di riferimento

Esistono moltissimi modelli di collaudo del software e alcuni standard specifici. Molti di questi sono modelli puramente teorici, altri molto più pratici. Alcuni definiscono i requisiti per le attività di verifica e validazione all'interno del sistema di gestione per la qualità, nell'ambito del processo di realizzazione di un prodotto. Altri ancora rappresentano dei modelli per il miglioramento delle prestazioni aziendali. Ognuno di questi ha la sua validità nel contesto in cui è collocato ed utilizzato. Qui si vogliono descrivere brevemente solo tipologie di essi: gli standard che indirizzano, direttamente o indirettamente, le attività di collaudo ed i modelli di maturità per il miglioramento delle prestazioni. I modelli e gli standard sono riportati per due ordini di motivi: primo perché ad alcuni di essi si fa esplicito riferimento la metodologia proposta, secondo per amore di completezza dell'argomento trattato. Il lettore può saltare la lettura del presente capitolo se non fosse interessato a ciò e ritornarvi in seguito, quando lo ritenesse necessario.

1.4.1 Standard relativi al Software Testing

Lo stato degli standard sul testing è in evoluzione e vede coinvolte le tre organizzazioni principali: ISO/IEC, BSI e IEEE.

Attualmente BSI mantiene due standard sul testing:

- **BS 7925-1**, Software Testing: Part 1 – Vocabulary;
- **BS 7925-2**, Software Testing: Part 2 – Software Component Testing.

IEEE mantiene altri due standard:

- **IEEE Std 829**, Software Test Documentation;
- **IEEE Std 1008**, Software Unit Testing.

ISO/IEC mantiene a sua volta importanti standard contestuali all'argomento:

- **ISO/IEC 12207**, Software Life Cycle Processes (Stabilisce un quadro di riferimento comune per i processi del ciclo di vita del software e contiene i processi, le attività ed i compiti che possono essere applicati durante l'approvvigionamento di prodotti e servizi software, lo sviluppo, la conduzione operativa, la manutenzione ed il ritiro dei prodotti software. I processi includono quelli relativi alla Verifiche e Validazioni (Testing);

- **ISO/IEC 15289**, System and Software Life Cycle Process Information Products (E' uno standard internazionale scritto per assistere gli utenti della norma ISO/IEC 12207 nella gestione degli elementi informativi prodotti durante il ciclo di vita del software. Stabilisce un quadro di riferimento comune per la descrizione del ciclo di vita dei sistemi, inclusa la documentazione di progetto e quella tecnica, tra cui, piani, report, specifiche relativi alle attività di verifica e validazione);
- **ISO/IEC TR 19759**, Guide to the Software Engineering Body of Knowledge (Definisce le 10 competenze richieste alla professione di ingegnere del software in base ad un consenso generale. Una competenza è proprio quella del Software Testing).

Esiste un progetto in corso con l'obiettivo di accorpare gli standard esistenti sul testing in un unico corpo di norme sull'argomento. Le tre organizzazioni coinvolte (SC 7 - il gruppo di lavoro dell'ISO in materia -, BSI e IEEE-CS) collaborano a tale scopo. Lo standard nuovo, con il titolo "Software Engineering – Software Testing", è pianificato avere quattro parti:

- **Part 1: Testing concepts** (conterrà una panoramica dei principi e delle pratiche di testing, i concetti ed il vocabolario, i principi utili a capire le altre tre parti della norma);
- **Part 2: Testing process** (conterrà la parte della norma ISO/IEC 12207:2007 relativa al processo di testing ed una guida ai requisiti minimi richiesti dalla norma);
- **Part 3: Test documentation** (conterrà i requisiti relativi alla documentazione di test);
- **Part 4: Test case design** (conterrà una guida alla progettazione dei casi di test).

I due aspetti più importanti del progetto sono: a) rendere disponibile di un unico corpo tutte le norme relative al testing; b) indirizzare l'intero ciclo di vita del testing, includendo il test unitario[1], il test di integrazione, quello di sistema, quello di qualificazione e quello di accettazione.

L'ISO detiene anche altre norme, oltre a quelle elencate sopra, che pur essendo di carattere generale hanno ugualmente aspetti inerenti il testing:

[1] Si ricorda che le norme attualmente disponili indirizzano solo il tipo di test unitario.

- **ISO/IEC 9000:2000** costituisce lo standard di riferimento per i sistemi di gestione per la qualità. Una parte specifica della norma è dedicata all'attività di verifica e validazione[2];
- **ISO/IEC 9126** costituisce lo standard sulla qualità del software;

Gli standard elencati sono brevemente descritti in appositi paragrafi nel seguito del capitolo.

1.4.2 Modelli relativi al Software Testing

Tra i modelli più conosciuti ricordiamo:

- **Capability Maturity Model**[3], inizialmente realizzato per lo sviluppo di prodotti software (SW/CMM), si è poi evoluto in un modello più generale che include la realizzazione di prodotti generici (CMMI). Tra i processi previsti dal modello ci sono quelli relativi al test (Verifica e Validazione);
- **Testing Maturity Model** (TMM), costituisce l'evoluzione del modello CMM applicato specificatamente al testing del software;
- **Software Engineering Body Of Knowledge** (SWEBOK) è il modello che definisce le competenze richieste ai professionisti del software; il capitolo "5 Software Testing" del modello è dedicato all'attività di collaudo del software. Il modello è stato recentemente pubblicato dall'ISO come Technical Report: ISO/IEC TR 19759:2005, Guide to the Software Engineering Body of Knowledge.

1.1.1 Definizioni

Anomalia, Difetto, Errore, Malfunzionamento

Nel linguaggio comune degli sviluppatori questi termini sono utilizzati, generalmente, in maniera indiscriminata per identificare un "problema" nel software. In termini più precisi, essi hanno significati diversi, anche se non è essenziale ai fini della comprensione della metodologia di test. Per amore di chiarezza e completezza segue una breve descrizione di ciascun termine.

- **Anomalia (*Fault*)** è un difetto del software. I due termini anomalia e difetto sono equivalenti, anche se spesso utilizzati impropriamente.

---

[2] L'attività di verifica ha lo scopo di garantire che si sta realizzando un "buon prodotto", mentre l'attività di validazione ha lo scopo di garantire che si sta realizzando il "prodotto giusto". La prima attività mira a garantire la qualità del prodotto; la seconda a garantire che esso indirizzi i requisiti e le specifiche definite per esso.

[3] La versione attuale del modello è il Capability Maturity Model Integration (CMMI) rilasciato dal Software Engineering Institute (SEI) di Pittsburgh, Pennsylvania (USA).

- **Difetto (*Fault*)** è un vizio nel software che genera un malfunzionamento quando esso venga eseguito. Ciò costituisce un "problema" per l'utente che non può utilizzare la funzionalità richiesta. Il termine è sinonimo di anomalia.
- **Errore (*Error, Bug*)** è la causa di un difetto. E' commesso a livello operativo in fase di sviluppo scrivendo del codice sorgente non corretto oppure a livello concettuale interpretando non correttamente le specifiche o i requisiti. In entrambi i casi il sostare prodotto risulta avere un difetto e, quando sarà eseguito, produrrà come effetto un malfunzionamento.
- **Malfunzionamento (*Fault*)** è il comportamento del software in maniera non conforme alle specifiche. Il software, cioè, non si comporta come ci si aspetta che faccia.

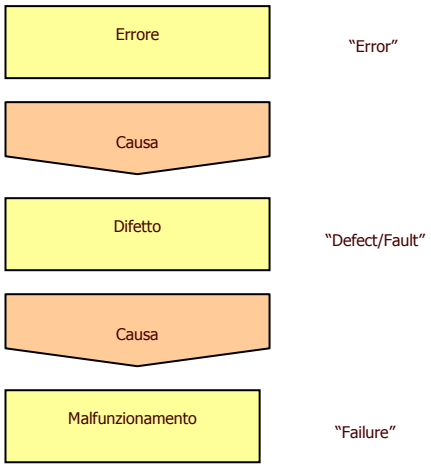

Figura 1. Relazione causa-effetto nel software.

Riassumendo, occorre distinguere tra la causa di un malfunzionamento e gli effetti del malfunzionamento stesso. Per la prima (causa) si parla di difetto generato, a sua volta da un errore nel codice sorgente; per il secondo (effetto) si parla di malfunzionamento. L'attività di testing evidenzia i malfunzionamenti nell'esecuzione del software; la correzione dell'errore nel codice sorgente che ha causato il malfunzionamento è l'attività conclusiva dello sviluppo software.

Riesame, verifica e validazione

- **Riesame** rappresenta l'attività volta a garantire che il prodotto finale indirizzi correttamente e completamente i requisiti per esso definiti, espliciti ed impliciti, funzionali e qualitativi. L'attività è svolta tramite revisioni tecniche o ispezioni dei documenti prodotti nella fase in essame.
- **Verifica (*Verification*)** rappresenta l'attività volta a garantire che un determinato prodotto intermedio (prodotti di fase) sia coerente con le speci-

fiche definite per esso (input di fase). Per esempio, la revisione di un documento di disegno verifica che esso sia aderente alle specifiche dichiarate per esso - specifiche funzionali, requisiti tecnici, standard documentali e di progettazione, ecc. -. La revisione di un modulo software (codice sorgente), a sua volta, verifica che esso sia coerente con il suo disegno tecnico e con gli standard di programmazione stabiliti. L'attività di verifica è svolta tramite "ispezioni", "revisioni", ecc.

- **Validazione (*Validation*)** rappresenta l'attività volta a garantire che un prodotto indirizzi i requisiti originali stabiliti per esso quando venga utilizzato nell'ambiente cui è destinato. La validazione del software è svolta tramite attività di testing; eseguita in un ambiente uguale o simile a quello di produzione, la validazione del disegno può invece essere fatta tramite la valutazione di un prototipo che implementi la soluzione in condizioni simili a quelle effettive di utilizzo, ecc.

La figura che segue mostra le attività di verifica e di validazione nel ciclo di vita del software.

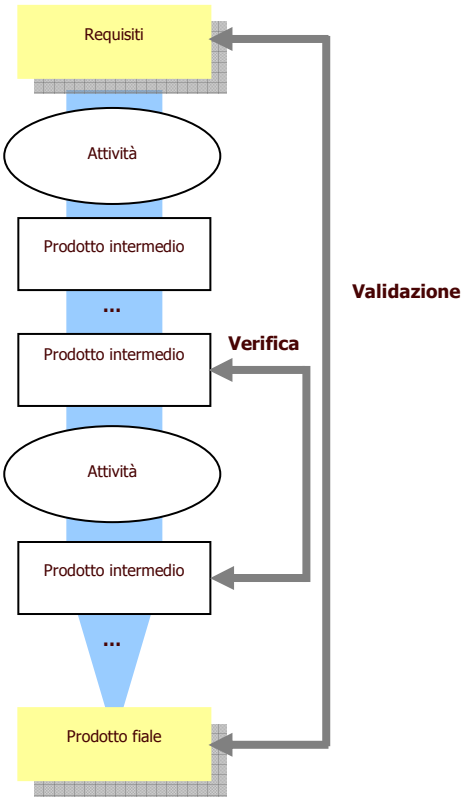

Figura 2. Verifica e validazione nel ciclo di vita.

Testing, risultato atteso, varianza

- **Testing** rappresenta il processo con il quale si esegue e si valuta, automaticamente o manualmente, un programma, un prodotto o un sistema per verificare se soddisfa i requisiti specificati e per identificare le differenze tra i risultati attesi e quelli ottenuti.
- **Risultato atteso**: Un insieme di requisiti o di specifiche cui devono conformarsi i risultati dei test ottenuti in output da un processo per essere accettati come validi. Un tipico esempio di risultato atteso è rappresentato dalle specifiche di una transazione on-line i cui tempi di risposta devono essere inferiori, per esempio, a 2 secondi.
- **Varianza** rappresenta la discrepanza osservata tra i risultati ottenuti e quelli attesi durante i test. Può essere causata da diversi motivi: errori nel software, errori nella definizione dei risultati attesi, utilizzo di dati invalidi, ecc.

Controllo, revisione, test, collaudo, certificazione

- **Controllo** è l'insieme delle attività svolte per verificare la conformità di un prodotto, servizio o processo ai suoi requisiti. Rispetto ai rapporti tra cliente e fornitore si parla di controllo "interno" quando questo sia svolto dal fornitore stesso o di controllo "esterno" quando esso sia svolto direttamente dal committente o da una terza parte da esso designata. Nell'ambito della gestione della qualità il controllo è detto "assicurazione qualità". Nell'ambito del software testing il controllo è detto "statico" o "dinamico" a seconda che le attività di verifica siano svolte in maniera automatica esercitando il software (in gergo "facendo girare" il software) oppure manualmente tramite revisioni tecniche dei documenti o del codice stesso.
- **Revisione** è l'attività manuale svolta da personale tecnico (programmatori, analisti, progettisti) su documenti tecnici di progetto (requisiti, specifiche, disegno, casi di test, ecc.) per verificarne la completezza, la correttezza e la conformità a standard e requisiti. Lo scopo principale è quello di scoprire gli eventuali errori presenti nei documenti sottoposti a revisione e permetterne la correzione. La revisione può essere condotta anche sul codice sorgente inteso come una forma di documento. Una revisione può essere condotte in maniera "formale" oppure "informale". Nel primo caso deve seguire una prassi definita in azienda che prevede di formalizzarne lo svolgimento ed i risultati. Nel secondo caso non è richiesta alcuna formalizzazione. I termini comunemente usati per le revisioni sono: "ispezione" e "walkthrough". **Ispezione** identifica la revisione tecnica di un documento o di un codice sorgente. **Walkthrough** è una tecnica di revisione in cui

i partecipanti simulano a tavolino il comportamento del sistema (il software) per controllarne la correttezza. **Peer review** è invece il termine che definisce una modalità di svolgimento di una revisione in cui i revisori sono "alla pari" senza alcuna gerarchia. Nell'ambito dei sistemi di gestione per la qualità l'ispezione è detta **verifica ispettiva** ed è svolta da una parte indipendente (interna o esterna) per verificare che le attività svolte siano conformi a quanto stabilito dal sistema stesso e siano l'efficaci ai fini del raggiungimento degli obiettivi stabiliti. In questo ambito lo standard ISO/IEC 10111 definisce i requisiti per lo svolgimento delle verifiche ispettive.

- **Test** (vedi **Testing**)
- **Collaudo** è la "validazione" del prodotto per verificare che rispetti i requisiti del committente. Nella legislazione italiana è definito come il controllo definitivo effettuato dal committente o da una terza parte da esso incaricata sul prodotto finale secondo le modalità indicate nel contratto che regola la commessa. Generalmente è detto collaudo di accettazione.
- **Certificazione** è una forma di "controllo" di un prodotto o di un processo per verificarne l'aderenza ad uno standard di riferimento e rilasciare, se prevista, la relativa certificazione di conformità. Nell'ambito del controllo della qualità si parla, ad esempio, della certificazione del sistema di gestione per la qualità alle norme ISO/IEC 9001:2000. In questo caso la certificazione attesta che il sistema qualità del fornitore, ed i processi produttivi inclusi nel sistema, sono conformi ai requisiti dello standard della serie ISO 9000. Si tratta quindi di un controllo "a priori" sui processi di sviluppo del software piuttosto che sul prodotto realizzato.

Gestione della qualità

- **Qualità** è l'insieme delle caratteristiche di un prodotto software che determinano la sua capacità a soddisfare le esigenze per cui è stato realizzato. In termini quantitativi il **livello di qualità** del prodotto software misura il grado di aderenza ai requisiti, espliciti ed impliciti. Le caratteristiche del software sono definite dalle norme ISO/IEC 9126 come: funzionalità, affidabilità, usabilità, efficienza, manutenibilità, portabilità. Ciascuna di esse ha propri attributi.
- **Sistema qualità** rappresenta l'insieme delle politiche, delle risorse e dei processi che un'organizzazione dedica in maniera esplicita per conseguire la qualità dei prodotti realizzati e dei servizi erogati. E' in pratica la garanzia che un'organizzazione offre ai clienti della qualità dei propri prodotti e servizi. Il **Manuale della qualità** descrive il sistema qualità stesso, mentre il **Piano della qualità** ne descrive le procedure messe in atto e le risorse rese disponibili per garantire il raggiungimento degli obiettivi di qualità stabiliti.

- **Pianificazione e controllo della qualità** è l'insieme delle attività di pianificazione e controllo svolte a sviluppare la qualità dei prodotti e dei servizi attesa dai committenti. La pianificazione è formalizzata nel **Piano della qualità del progetto** che è la descrizione delle procedure e delle risorse previste per garantire il raggiungimento degli obiettivi di qualità stabiliti per il progetto software. Esso fa riferimento al sistema qualità del fornitore.
- **Assicurazione qualità (**o **garanzia della qualità)** è l'insieme delle attività pianificate e svolte nell'ambito di un sistema qualità per assicurare che il fornitore soddisfi i requisiti del committente. In un'organizzazione software, per esempio, l'assicurazione qualità è svolta tramite verifiche ispettive volta a controllare che i progetti siano realizzati seguendo tutti i processi stabiliti dal sistema qualità aziendale.
- **Controllo della qualità** è l'insieme delle attività messe in atto in un progetto software per soddisfare i requisiti per la qualità. Nell'ambito dello sviluppo software, per esempio, le ispezioni ed i test sono attività di controllo della qualità del software.

1.1.2 Acronini

AQ	Assicurazione Qualità (QA = Quality Assurance)
CCM	Change and Configuration Management
CMM	Capability Maturity Model©
CMMI	Capability Maturity Model® Integration
CT	Caso di test (TC = Test case)
CU	Caso d'uso (UC = Use case)
DCR	Design Change Request
FP	Function Point (Punto funzione)
IEC	International Electrical Commission
ISO	International Organization of Standardization
IT	Integration Test (test d'integrazione)
LOC	Line od code (Linea di codice)
JCL	Job Control Language
MT	Matrice di test
ODC	Orthogonal Defect Classification
PM	Project Manager

PMBOK Project Management Body Of Knowledge

PMI Piccole e Medie Imprese
Project Management Institute

KLOC Kilo Line of code (Migliaia di linee di codice)

SAL Stato Avanzamento Lavori

SCM Software Configuration Management

SEI Software Engineering Institute

SMART Specifico, Misurabile, Attendibile, Realistico, Tempistico (riferito ad un requisito)

ST System Test (test di sistema)

SWEBOK Software Engineering Body Of Knowledge

TMM Testing Maturity Model

TR Technical Report

UAT User Acceptance Test (test di accettazione dell'utente)

UT Unit Test (test unitario)

2. Standard di riferimento

Il presente capitolo fornisce una panoramica degli standard di mercato e sui modelli di maturità che affrontano il tema oggetto di questo manuale: la qualità del software, il test e collaudo ed il controllo qualità.

Standard e modelli sono presi come riferimento nel manuale ed i relativi requisiti, best practice e raccomandazioni sono indirizzati nella metodologia proposta.

Chi non ha un interesse immediato ad approfondire i contenuti degli standard e dei modelli di maturità può non leggere il presente capitolo e saltare a quelli successivi.

2.1 IEEE Std 829-1998, Standard for Software Testing Documentation

In attesa che lo standard IEEE 829 confluisca nel nuovo corpo unico di norme relative al testing, descriviamo qui brevemente i contenuti di tale standard, documento di riferimento negli USA e non solo. Lo scopo è quello di definire una base per la creazione della documentazione relativa al testing. I documenti indirizzati dallo standard coprono la pianificazione dei test (*Test Planning*), le specifiche dei test (*Test Specification*) e la produzione di reportistica sul test (*Test Reporting*).

Il **piano di test** richiede che vengano indirizzati i seguenti argomenti: ambito ed obiettivi del test, approccio al testing, risorse previste, attività e relativa tempistica. Esso identifica inoltre gli elementi da sottoporre a test, le funzioni e le

caratteristiche (*features*) da testare, i compiti (*task*) da svolgere, il personale responsabile per ciascun compito, i rischi associati al piano.

Le **specifiche** relative al testing sono coperte da tre tipi di documenti:

- "Specifica per la progettazione dei test" - Raffina i concetti espressi nell'approccio al test fornendo i dettagli necessari ad individuare i componenti e le caratteristiche del software da indirizzare con la progettazione ed i relativi test da effettuare; identifica i casi di test e le procedure di test necessari per completare la fase di testing e definisce i criteri di completamento/fallimento dei test.
- "Specifica per il documento dei casi di test" – Specifica i dati di input da fornire al sistema per ciascun caso di test e il relativo output atteso. Identifica anche le condizioni per eseguire le procedure stabilite per ciascun caso di test. I casi di test sono documentai separatamente dalla progettazione in quanto il primo (design) definisce le condizioni generali del caso di test mentre il secondo (caso di test singolo) specifica i dati di input e di output da utilizzare nell'esecuzione. Un caso di test può essere utilizzato successivamente in condizioni diverse cambiando solo i dati di input.
- "Specifica per la procedura di test" – Identifica tutti i compiti da eseguire per completare un caso di test secondo la progettazione effettuata. Le procedure di test sono generalmente descritte in un documento separato dal caso di test per consentire che siano eseguite passo dopo passo senza alcun dettaglio su informazioni estranee ad esse.

La **reportistica** di test è coperta da quattro documenti:

- "Rapporto di trasmissione degli elementi di test" – Identifica i componenti software resi disponibili per il testing quando: (1) i due gruppi di lavoro relativi allo sviluppo ed al testing siano separati e richiedano una comunicazione formale, (2) per formalizzare l'avvio delle attività di testing.
- "Registrazione dei test (Test Log)" – E' utilizzato dal gruppo di test per registrare quanto avviene durante l'esecuzione dei casi di test.
- "Rapporto di incidente di test" – Descrive ogni situazione anomala rilevata durante l'esecuzione dei test e che richieda un'analisi dell'accaduto.
- "Sommario di test" – Riassume le attività di testing eseguite a fronte delle specifiche di progettazione dei test.

La figura che segue riassume la documentazione coinvolta nell'attività di testing e la loro relazione.

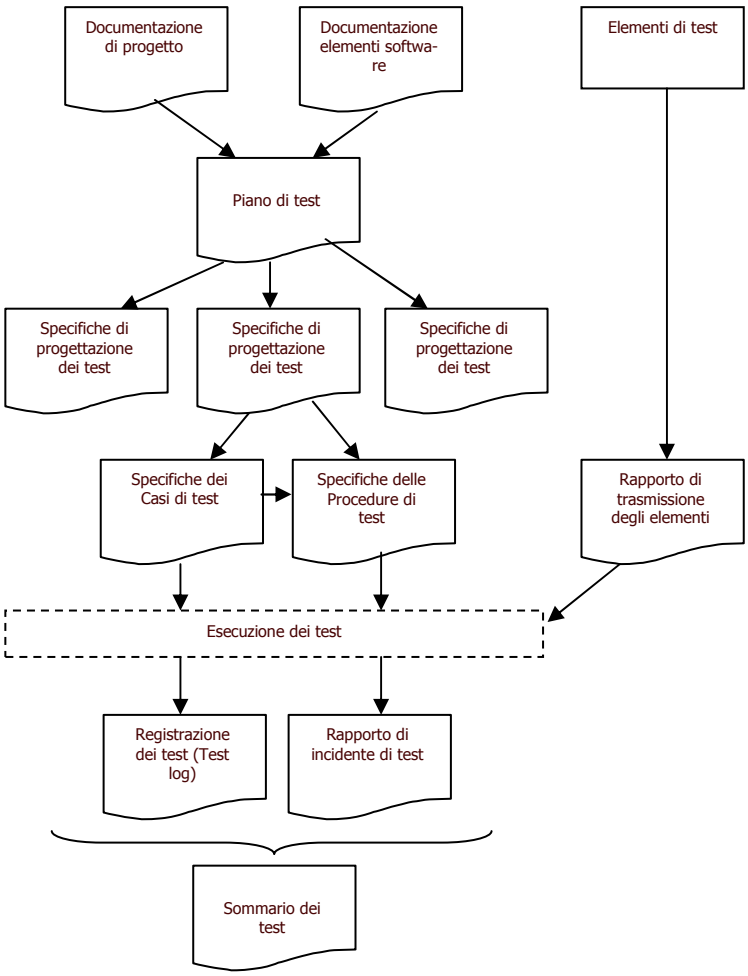

Figura 3. Relazione tra i documenti di testing e processo di test.

> *La presente metodologia prende come riferimento quanto descritto nello standard e lo adatta alle necessità di semplificazione e coerenza dei progetti di medie dimensioni. Per esempio, oltre al piano di test, è previsto un unico documento che descrive la progettazione dei casi, i casi di test in dettaglio e la le relative procedure. Inoltre, i contenuti previsti dallo standard per ciascun documento sono presi come riferimento dalla metodologia presente ma gli schemi ed i modelli presentati nel capito apposito ("Capitolo 11. Produzione dei documenti di test") sono opportunamente semplificati per adattarsi alle caratteristiche reali dei progetti di medie e piccole dimensioni. La semplificazione è fatta in base all'esperienza maturata dall'autore proprio nei gruppi di test di organizzazioni software.*

2.1 ISO 9000, Sistemi di gestione per la qualità

2.1.1 Generalità

Lo standard ISO 9000 attuale (Vision 2000) ha compiuto un passo notevole verso il miglioramento continuo della qualità. Un'azienda software trova ora quanto necessario per portare la propria organizzazione verso un miglioramento concreto e tangibile. L'attuale Vision 2000 costituisce un modello concreto di "miglioramento continuo".

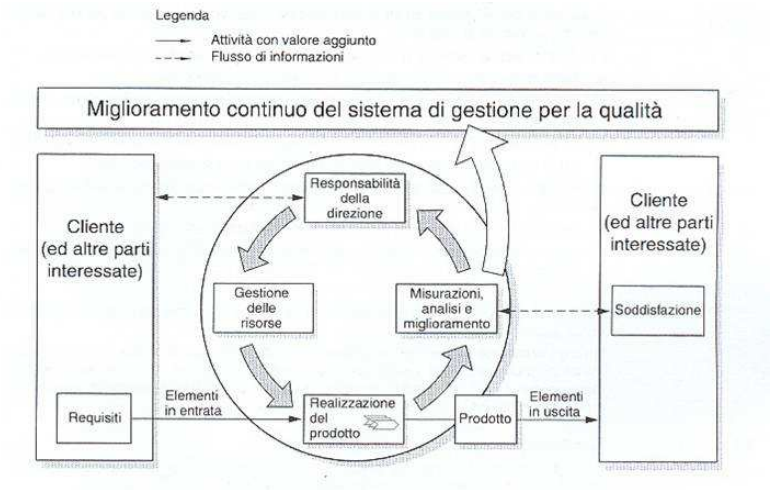

Figura 4. Schema ISO9000 di miglioramento continuo.

Occorre però evitare di costruire cattedrali nel deserto.

Sistemi di qualità perfettamente aderenti alle norme ma poco utili dal punto di vista produttivo servono a poco o niente. Sistemi realizzati solo per risultare "conformi" (solo per ottenere il "bollino blu") sono molto costosi e non producono risultati economici che giustifichino gli investimenti fatti. Occorre andare alla sostanza, semplificare, condividere e poi ... agire. Con competenza, continuità e coerenza!

L'Aicq (Associazione italiana per la cultura della qualità), con il suo comitato per la qualità del software, ha realizzato un utilissimo quaderno per le organizzazioni software che descrive l'interpretazione delle norme e si suggeriscono modalità per la loro applicazione.

Il quadro normativo attuale è stato notevolmente semplificato ed è costituito da quattro documenti principali:

ISO 9000 Fondamenti e terminologia, descrive i fondamenti dei sistemi di gestione per la qualità e ne specifica la terminologia adoperata;

ISO 9001 Requisiti per i sistemi di gestione per la qualità, specifica i requisiti dei sistemi di gestione per la qualità che un'azienda deve soddisfare per dimostrare ai propri clienti, ed al mercato in generale, la propria capacità di fornire prodotti e servizi in linea con la qualità attesa;

ISO 9004 Guida al miglioramento continuo, fornisce linee guida per adottare le norme ISO 9000 in maniera proficua, migliorando le prestazioni aziendali, sia in termini di efficacia che di efficienza, in modo continuativo (miglioramento continuo);

2.1.2 La norma e le verifiche e validazioni

Vediamo ora cosa dice la norma ISO 9001 a proposito del test e collaudo.

Il punto della norma "7. Realizzazione del prodotto" indirizza la pianificazione, la progettazione, la realizzazione ed il collaudo del prodotto. In particolare, per le attività di verifica e validazione sono previsti i seguenti punti specifici.

> **7. Realizzazione del prodotto**
>
> ……..
>
> **7.3.4 Riesame della progettazione e dello sviluppo**
>
> In fasi opportune devono essere effettuati riesami sistematici della progettazione e dello sviluppo, in accordo con quanto pianificato, al fine di:
>
> - valutare la capacità dei risultati della progettazione e dello sviluppo di ottemperare ai requisiti,
> - individuare tutti i problemi e proporre le azioni necessarie.
>
> A tali riesami devono partecipare rappresentati delle funzioni coinvolte nelle fasi di progettazione e di sviluppo oggetto del riesame. Le registrazioni dei risultati dei riesami e delle eventuali azioni necessarie devono essere conservate.
>
> **7.3.5 Verifica della progettazione e dello sviluppo**
>
> Devono essere effettuate verifiche, in accordo con quanto pianificato, per assicurare che gli elementi in uscita dalla progettazione e dallo sviluppo siano compatibili con i relativi requisiti in ingresso. Le registrazioni dei risultati delle verifiche e delle eventuali azioni necessarie devono essere conservate.
>
> **7.3.6 Validazione della progettazione e dello sviluppo**
>
> Deve essere effettuata la validazione dalla progettazione e dallo sviluppo in accordo con quanto pianificato, per assicurare che il prodotto risultante dalla progettazione e dallo sviluppo sia in grado di soddisfare i relativi requisiti per l'applicazione specificata o, dove conosciuta, per quella prevista. Dove applicabile, la validazione deve essere completata prima della consegna o dell'utilizzazione del prodotto. Le registrazioni dei risultati della validazione e delle eventuali azioni necessarie devono essere conservate.

7.3.7 Tenuta sotto controllo delle modifiche della progettazione e dello sviluppo

Le modifiche della progettazione e dello sviluppo devono essere identificate e le relative registrazioni conservate. Le modifiche devono essere riesaminate, verificate e validate, come opportuno, ed approvate prima della loro attuazione. Il riesame delle modifiche della progettazione e dello sviluppo deve comprendere la valutazione degli effetti che tali modifiche hanno sulle parti componenti e sui prodotti già consegnati. Le registrazioni dei risultati delle modifiche e delle eventuali azioni necessarie devono essere conservate.

2.2 ISO/IEC 9126 Modello per la qualità del software

Le norme ISO 9126 rappresentano il modello per la qualità del software e definiscono le caratteristiche e gli attributi del software[4].

Il modello è composto da quattro norme descritte in altrettanti documenti.

La prima (ISO/IEC 9126-1) definisce il modello per la qualità dei prodotti software.

Le altre tre (ISO/IEC TR 9126-2, 3, 4) partono dal modello appena menzionato e definiscono le caratteristiche e le relative metriche per la qualità "esterna", "interna" ed "in uso".

Le caratteristiche della qualità dei prodotti software sono definite come: Funzionalità, Affidabilità, Usabilità, Efficienza, Manutenibilità, Portabilità.

Per ogni caratteristica sono definite sotto-caratteristiche (attributi) che identificano i diversi componenti del software.

L'insieme di tutte le caratteristiche ed i relativi attributi costituiscono il modello per la qualità interna, esterna ed in uso dei prodotti software.

Nel dettaglio, le norme in oggetto sono:

- ISO/IEC 9126-1:2001, Software Engineering - Product Quality Part 1: Quality Model;
- ISO/IEC TR 9126-2:2003, Software Engineering - Product Quality Part 2: External Metrics;
- ISO/IEC TR 9126-3:2003, Software Engineering - Product Quality Part 3: Internal Metrics;

[4] Maggiori dettagli sul modello e sulla sua applicazione pratica nei progetti di sviluppo software sono forniti in un apposito documento dello stesso autore (La qualità del software secondo il modello ISO/IEC 9126) disponibile nel suo sito (www.colonese.it/pubblicazioni).

- ISO/IEC TR 9126-4:2004, Software Engineering - Product Quality Part 4: Quality in Use Metrics.

Di seguito sono descritte le caratteristiche menzionate ed i relativi attributi.

Funzionalità

Definisce la ricchezza delle capacità funzionali del software nel soddisfare i requisiti, espliciti ed impliciti. "Il software fa quello che ci si aspetta debba fare".

I suoi attributi principali da prendere in considerazione sono:

- Completezza (delle funzioni offerte verso i requisiti);
- Accuratezza (con la quale le funzioni sono offerte);
- Interoperabilità (con gli altri sistemi presenti nell'ambiente nel quale la soluzione opererà);
- Aderenza (a normative, leggi, regole, standard, ecc.);
- Sicurezza (dei dati e delle persone).

Affidabilità

Definisce la capacità del software di mantenere i livelli di prestazione stabiliti nell'ambito di condizioni definite e per una durata stabilita. "Il software reagisce positivamente alle variazioni dell'ambiente circostante".

I suoi attributi principali da prendere in considerazione sono:

- Maturità (livello di assestamento del sistema);
- Tolleranza ai guasti (margini entro i quali le prestazioni sono fornite);
- Recuperabilità (capacità di ripristinare la situazione originale in caso di caduta o degrado).

Usabilità

Definisce l'impegno richiesto per usare il software da parte degli utenti stabiliti. "Il software gestisce l'interazione con gli utenti in modo ottimale" .

I suoi attributi principali da prendere in considerazione sono:

- Comprensibilità (facilità di capire e intuire le funzionalità disponibili nel sistema);
- Apprendibilità (facilità di imparare e ricordare le funzionalità offerte);
- Operabilità (facilità di adoperare le funzioni rese disponibili).

Efficienza

Definisce la capacità di erogare le funzioni offerte con il minor uso di risorse impiegate, in condizioni di funzionamento stabilite. Spesso è detta "performance". "Il software usa bene le risorse disponibili" .

I suoi attributi principali da prendere in considerazione sono:

- Rispetto al tempo (tempo di risposta);
- Rispetto alle risorse (utilizzo di risorse).

Manutenibilità

Definisce la facilità con cui è possibile eseguire modifiche al software esistente intese come evoluzioni, correzioni, adattamenti, ristrutturazioni, ecc. "Il software segue l'evoluzione dell'organizzazione" .

I suoi attributi principali da prendere in considerazione sono:

- Analizzabilità (facilità a diagnosticare i problemi, detta anche "problem determination");
- Modificabilità (facilità ad inserire nel software esistente le modifiche richieste);
- Stabilità (capacità del software esistente a rimanere stabile anche dopo le modifiche);
- Provabilità (facilità di eseguire test alle modifiche coinvolgendo poco le funzioni esistenti).

Portabilità

Definisce la facilità con la quale il software può essere portato da una piattaforma ad un'altra. "Il software segue l'evoluzione tecnologica" .

I suoi attributi principali da prendere in considerazione sono:

- Adattabilità (al nuovo sistema);
- Installabilità (facilità di installazione nel nuovo sistema) ;
- Conformità (alle regole del nuovo sistema);
- Sostituibilità.

Questi elementi sono presi in considerazione quando si decide la strategia di test (quali caratteristiche del software occorre indirizzare ed in quale modo). Le stesse caratteristiche ed i relativi attributi sono poi presi in considerazione durante la progettazione dei test, la predisposizione degli ambienti di test e la realizzazione delle basi di dati necessari per eseguire i test.

2.3 ISO/IEC 12207 Ciclo di vita del software

Si tratta di una famiglia di norme che indirizzano i processi di produzione ed il ciclo di vita del software. Esse definiscono un modello base per i processi del ciclo di vita del software secondo l'approccio più moderno. Sono norme generali e devono essere personalizzate alle esigenze della propria organizzazione.

Sono fornite anche delle linee guida per il loro utilizzo nei progetti di sviluppo software.

- ISO/IEC 12207:1995, Information Technology - Software Life Cycle Processes; alla norma si sono aggiunte negli ultimi anni due modifiche che ne aggiornavano i contenuti: Amendment 1: 2002 e Amendment 2:2004;

- ISO/IEC TR 15271: 1998, Information Technology - Guide for ISO/IEC 12207 (Software Life cycle Process);
- ISO/IEC TR 16326:1999, Software Engineering - Guide for the Application of ISO/IEC 12207 to Project Management.

2.3.1 Principali differenze con ISO 9001

In sintesi le principali differenze fra ISO 9001 e ISO 12207 sono le seguenti:

ISO 9001	ISO 12207
• Si applica ad organizzazioni di qualsiasi genere, dimensione e tipo di prodotti forniti; • Si occupa della qualità dal punto di vista dell'azienda; • Definisce specifiche attività di gestione; • E' utilizzata per la certificazione del sistema qualità del fornitore.	• Si applica ad organizzazioni di produzione e di servizio nel campo del software; • Si concentra sul ciclo di vita del software; • Si occupa di tutte le attività del ciclo di vita del software, sia gestionali che tecniche; • E' utilizzata per acquisire, fornire, sviluppare, utilizzare e mantenere un prodotto software, e non prevede procedure per la valutazione del fornitore.

2.3.2 Contenuti della norma ISO/IEC 12207

La norma, emessa nel 1995 ed aggiornata nel 2002 , ha lo scopo principale di definire in modo preciso e condiviso i processi del Ciclo di Vita del Software: dalla formulazione dei requisiti, allo sviluppo, all'esercizio ed alla manutenzione.

I processi produttivi e gestionali delle organizzazioni sono suddivisi dalla norma in tre categorie:

1. **primari**, comprendenti le attività direttamente legate allo sviluppo del software,
2. **di supporto**, che includono la gestione dei documenti e dei processi di controllo della qualità,
3. **organizzativi**, che coprono gli aspetti manageriali e di gestione delle risorse.

Per ciascun processo la norma evidenzia chiaramente:

- obiettivo e responsabilità,
- lista delle attività che lo compongono,
- singoli compiti nei quali è suddivisa ogni attività.

La norma fa proprio il ciclo di miglioramento dei processi basato sulla sequenza Plan-Do-Check-Act (pianifica, esegui, controlla, attua eventuali modifiche)[5].

[5] L'Aicq-ci ha realizzato un apposito quaderno che tratta l'argomento e fornisce validi suggerimenti su come applicare la norma ISO 12207 nelle organizzazioni software.

1. Processi primari

Essi indirizzano le seguenti attività principali:

- Acquisizione,
- Fornitura,
- Sviluppo,
- Esercizio,
- Manutenzione.

Ciascun processo primario è definito e descritto dalla norma in termini di attività (*activities*) e compiti (*tasks*).

Ogni compito (*task*), a sua volta, indica cosa occorre fare ("what to do") e non "come" la si deve fare ("how to do")[6].

2. Processi di supporto

I processi di supporto aiutano le attività di tutti gli altri processi dell'organizzazione a garantire il successo e la qualità del progetto.

Essi indirizzano le seguenti attività principali:

- Gestione della documentazione;
- Gestione della configurazione;
- Assicurazione della qualità
- Verifica;
- Validazione;
- Revisioni congiunte;
- Audit;
- Risoluzione dei problemi.

La nostra metodologia di test è quindi direttamente interessata alle attività di **Verifica**, **Validazione**, **Revisioni congiunte** e **Risoluzione dei problemi**.

"Quaderno N. 17. Settembre 2004. Modello dei processi ISO 9000 applicato in aziende di sviluppo software e fornitura di servizi IT. Aicq-ci"

[6] La norma fa uso dei seguenti "verbi" per indicare i diversi livelli di obbligatorietà:

- dovrà, per la definizione dei requisiti;
- dovrebbe, per specificare le raccomandazioni;
- può, per indicare il permesso a fare qualcosa, o il fatto di essere in grado di farla;
- qualunque altro (tempo presente), per descrivere un preambolo o per descrivere il contesto.

3. Processi organizzativi

Tali processi eseguono funzioni a livello organizzativo aziendale, per supportare altri processi primari, di supporto o organizzativi. I processi organizzativi aiutano nel definire, controllare e migliorare gli altri processi.

Essi indirizzano le seguenti attività di gestionali:

- Gestione dello sviluppo,
- Gestione delle infrastrutture,
- Gestione del miglioramento,
- Formazione.

Per una descrizione completa e dettagliata della norma occorre fare riferimento ala norma stessa.

2.3.3 ISO/IEC TR 15271, Guida all'utilizzo della norma ISO/IEC 12207 sul ciclo di vita del software

Si tratta di un rapporto tecnico (TR) emesso nel 1998 con lo scopo di fornire una guida all'applicazione della norma ISO/IEC 12207 relativamente al ciclo di vita del software. Esso tiene conto dei vari fattori da considerare quando si applica la norma nei diversi cicli di vita del software delle organizzazioni.

La guida non fornisce quindi nuovi requisiti da applicare ai cicli di vita ma piuttosto consiglia come interpretare ed applicare quelli (i requisiti) già espressi dalla norma ISO/IEC 12207.

La guida discute tre modelli fondamentali di cicli di vita e fornisce esempi di come personalizzarli nelle diverse realtà.

Maggiori dettagli sui contenuti della norma si possono ottenere direttamente dalla sua lettura.

2.3.4 ISO/IEC 15289:2008, System and software engineering – Content of system and software life cycle process information products (Documentation)

La norma è stata sviluppata per assistere gli utenti dei processi contenuti nel ciclo di vita dei sistemi e del software. In particolare, assiste nella gestione degli elementi informativi (documentazione). Si basa sui processi definiti dal ciclo di vita come specificato nella norma ISO/IEC 12207. Gli elementi informativi sono essenziali ai fini della gestione dei processi produttivi e gestionali e rappresentano "documenti da consegnare (deliverable)".

La norma identifica tutti gli elementi informativi richiesti dai processi contenuti nel ciclo di vita del software e ne definisce lo scopo ed i contenuti. I contenuti, in particolare, sono definiti in base a generici tipi di documenti (es.: piani, specifiche, rapporti, ecc.).

Quanto specificato dalla norma è applicabile a tutti i progetti, prodotti e sistemi software indipendentemente dalla loro complessità e dimensione.

2.4 ISO/IEC TR 19759 IEEE, Guide to the Software Engineering Body Of Knowledge (SWEBOK)

Il Software Engineering Body of Knowledge (SWEBOK) rappresenta un importante passo verso la definizione delle competenze richieste ai professionisti del settore. Le competenze descritte sono quelle "core" su cui si è avuto il consenso di tutte le parti coinvolte nel progetto[7]. Oltre alle competenze, il progetto ha definito un documento con le regole etiche per la professione.

Lo studio definisce 10 competenze di base per la professione di Ingegnere del software:

1. Software requirements
2. Software design
3. Software construction
4. Software testing
5. Software maintenance
6. Software configuration management
7. Software engineering management
8. Software engineering process
9. Software engineering tools and methods
10. Software quality

Ciascuna competenza è descritta in un apposito capitolo.

Il capitolo 10 del rapporto tecnico è dedicato al testing del software.

Il test del software consiste nella verifica dinamica del comportamento del software realizzato tramite un numero finito di casi di prova selezionati tra un numero pressoché infinito di modalità diverse di utilizzo e a fronte di un comportamento specificato ed atteso. Sono definite cinque sotto-aree di competenza:

- Basic concepts
- Test levels

[7] Nel progetto sono praticamente rappresentate tutte le organizzazioni interessate al software, pubbliche e private, di tutto il mondo, inclusa l'Italia.

- Test techniques
- Test-related measures
- Managing the test process

La prima sotto-area "Basic concepts" presenta la terminologia del testing, i fondamenti teorici del testing e le relazioni con le altre attività del ciclo di vita del software.

La seconda sotto-area "Test levels" indirizza i target e gli obiettivi dei diversi test.

La terza sotto-area "Test techniques" descrive due categorie di tecniche: la prima categoria raggruppa le tecniche basate sui criteri con cui i test sono generati, la seconda categoria raggruppa le tecniche normalmente ignorate in fase di implementazione. Sono forniti suggerimenti su come selezionare e combinare le diverse tecniche in base ai target ed agli obiettivi da raggiungere.

La quarta sotto-area "Test-related measures" divide le misure tra quelle per valutare il software testato e quelle per valutare l'efficacia dei test eseguiti.

La quinta ed ultima sotto-area "Managing the test process" descrive le attività principali e gli elementi da prendere in considerazione.

La descrizione dei singoli temi è chiara ed esaustiva con numerosissimi riferimenti a pubblicazioni esterne specializzate sui singoli temi.

2.5 Modelli di maturità

2.5.1 Capability Maturity Model Integration (CMMI)

Il CMMI è un modello per il miglioramento dei processi di sviluppo prodotti e servizi. Consiste di "best practice" che indirizzano le attività di sviluppo e manutenzione e che coprono l'intero ciclo di vita, dal concepimento del prodotto al suo rilascio sul mercato e in produzione presso i clienti.

Realizzato dal Software Engineering Insitute (SEI), il modello di processo da indicazioni di quanto debba essere fatto per raggiungere determinati livelli di maturità[8] in un'organizzazione (nel nostro caso in un'organizzazione che sviluppa software).

Il modello si basa su cinque livelli di maturità crescenti (da 1 a 5).

[8] In realtà il modello definisce due rappresentazioni diverse dello stesso modello: una basata su 6 Capability Levels (0 – 5) ed una basata su 5 Maturity Levels (1 – 5). Le due rappresentazioni sono equivalenti, anche se formalmente distinte. La seconda, forse più diffusa è quella descritta in questo capitolo.

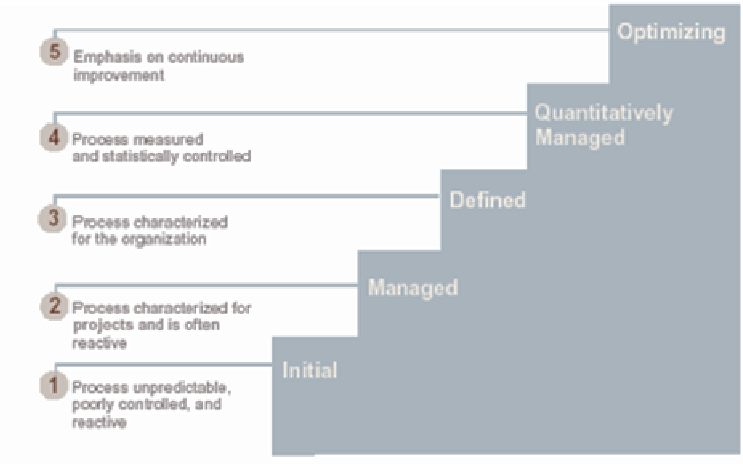

Figura 5. Schema dei livelli di maturità del modello CMMI (tratto dal sito CMMI del SEI).

L'attuale versione del modello (v1.2) definisce una serie di sottoprocessi (dette Aree di processo – KPA) da implementare a seconda del livello di maturità da raggiungere.

Livello di maturità 1: Eseguito. Nel primo livello in realtà non esistono processi definiti completamente o comunque questi non sono seguiti nella realizzazione dei progetti. La realizzazione è fatta in base alle competenze dei singoli ed i risultati, che in alcuni casi possono essere anche ottimi, non sono ripetibili in altri progetti realizzati da altre persone.

Livello di maturità 2: Gestito. Nel secondo livello si propongono i processi tipici gestionali: Gestione dei requisiti, Pianificazione del progetto, Monitoraggio e controllo del progetto, Gestione dei fornitori, Valutazione e analisi, Assicurazione qualità per i prodotti e per i processi, Gestione della configurazione.

Livello di maturità 3: Definito. Nel terzo livello si passa alla standardizzazione dei processi. Tutti i progetti utilizzano gli stessi processi che sono adattati alle diverse esigenze dei singoli progetti in base a regole stabilite. Le aree di processo indirizzate nell'attuale versione del modello sono ben 14, tra cui i processi di verifica e validazione (testing): Sviluppo dei requisiti; Soluzione tecnica; Integrazione del prodotto; ***Verifica***; ***Validazione***; Coinvolgimento nel processo aziendale; Definizione del processo aziendale; Addestramento; Gestione integrata del progetto; Gestione integrata dei fornitori; Valutazione dei rischi; Analisi e risoluzione delle decisioni; Ambiente aziendale per l'integrazione; Definizione integrata del team di sviluppo.

Livello di maturità 4: Gestito quantitativamente. Il quarto livello prevede la gestione dei progetti in base a risultati quantitativi, oltre che qualitativi. Tutto è misurato ed i processi sono valutati in base ai risultati ottenuti. Le aree di processo indirizzate sono: Prestazione del processo aziendale; Gestione quantitativa del progetto.

Livello di maturità 5: Ottimizzato. Nell'ultimo livello i processi sono gestiti in ottica di miglioramento continuo. Le aree di processo indirizzate sono: Innovazione e spiegamento aziendale; Analisi e risoluzione causale.

Per ciascun livello di maturità sono definiti obiettivi generali ed obiettivi specifici da raggiungere. Inoltre sono proposte pratiche generiche e pratiche specifiche da seguire che possono essere interpretate come "best practices".

Esempio

A titolo di esempio si riporta quanto previsto dal modello per il processo di Verifica e per quello di Validazione, corrispondenti al processo di collaudo (test statico e test dinamico).

Il modello definisce obiettivi specifici (SG) e pratiche specifiche (SP) per ciascun processo. Definisce anche obiettivi generici (GG), validi per tutti i processi e pratiche generiche (GP) anch'esse valide per tutti i processi.

Essi sono elencate qui di seguito. Per ovvi motivi di semplificazione non viene fornito alcun dettaglio sui singoli elementi, consigliando il lettore di approfondire il tema direttamente tramite il modello.

Verification (VER)

"Scopo del processo di verifica è quello di assicurare che i prodotti intermedi (es. documento) indirizzino i requisiti specificati".

SG1 Preparare le verifiche

I prodotti intermedi sono sottoposti a verifica.

SP 1.1 Selezionare i prodotti intermedi da verificare

SP 1.2 Stabilire l'ambiente per le verifiche

SP 1.3 Stabilire le procedure ed i criteri per le verifiche

SG2 Eseguire le revisioni tecniche (Peer Review)

I prodotti intermedi selezionati sono sottoposti a revisione tecnica.

SP 2.1 Preparare le revisioni tecniche

SP 2.2 Condurre le revisioni tecniche

SP 2.3 Analizzare i risultati delle revisioni tecniche

SG3 Verificare i prodotti intermedi selezionati

I prodotti intermedi sono verificati a fronte dei requisiti specificati.

SP 3.1 Effettuare le verifiche

SP 3.2 Analizzare i risultati delle verifiche

Validation (VAL)

"Scopo del processo di validazione è quello di dimostrare che il prodotto indirizza i requisiti richiesti quando esso è collocato (utilizzato) nell'ambiente specificato".

Obiettivi specifici (SGn) e pratiche specifiche (SPn.m).

SG1 Preparazione per la validazione

Preparare la validazione del prodotto.

SP 1.1 Selezionare il prodotto da validare

SP 1.2 Stabilire l'ambiente per la validazione

SP 1.3 Stabilire le procedure ed i criteri per la validazione

SG2 Validare il prodotto

Il prodotto è validato per assicurare che esso sia adatto ad essere utilizzato nell'ambiente per cui è previsto.

SP 2.1 Effettuare la validazione

SP2.2 Analizzare i risultati della validazione

Per entrambi i processi valgono i seguenti obiettivi generici e pratiche generiche.

GG39 Istituzionalizzare il processo definito

GP 2.1 Stabilire la Politica aziendale (per i processi in questione)

GP 2.2 Pianificare il processo (sia di verifica che di validazione)

GP 2.3 Assicurare le risorse

GP 2.4 Assegnare le responsabilità

GP 2.5 Formare il personale

GP 2.6 Gestire la configurazione

GP 2.7 Identificare e coinvolgere tutte le parti interessate (stakehol ders)

GP 2.8 Monitorare e controllare il processo

GP 2.9 Valutare l'aderenza in maniera oggettiva

GP 2.10 Rivedere lo stato con la direzione

GP 3.1 Stabilire il processo definito

GP 3.2 Collezionare le informazioni necessarie per il miglioramento

Nello specifico, il modello suggerisce (o richiede) per ciascuna delle voci sopra citate una serie di particolari che consentono di seguire l'approccio più corretto.

La metodologia presentata in questo documento indirizza in maniera pratica tutti gli elementi definiti dal modello CMMI per i processi di verifica e validazione.

[9] La numerazione degli obiettivi e delle pratiche non è sequenziale in quanto essa si riferisce ad una notazione in cui sono presenti anche elementi che si riferiscono ad un'altra rappresentazione del modello, detta "continuous", diversa da quella presentata qui, detta "staged". Ma ciò ha poca importanza ai fini della discussione fatta nel nostro contesto.

2.5.2 Testing Maturity Model (TMM)

Gli autori[10] hanno definito un modello di maturità per il processo di testing (TMM) progettato per aiutare le organizzazioni che sviluppano software a valutare e migliorare il processo. Il modello TMM è complementare al modello CMM di cui riprende e indirizza in maniera più dettagliata gli elementi critici per il responsabile dei test (test manager), per gli specialisti di testing e per coloro che si occupano della qualità del software. Il processo di testing definito dal modello TMM indirizza in maniera più estesa tutte le attività inerenti la qualità del software. Gli autori dichiarano che l'utilizzo del modello nel processo di testing ha un impatto positivo sulla qualità del prodotto finale, sulla produttività e sul tempo di realizzazione dei progetti.

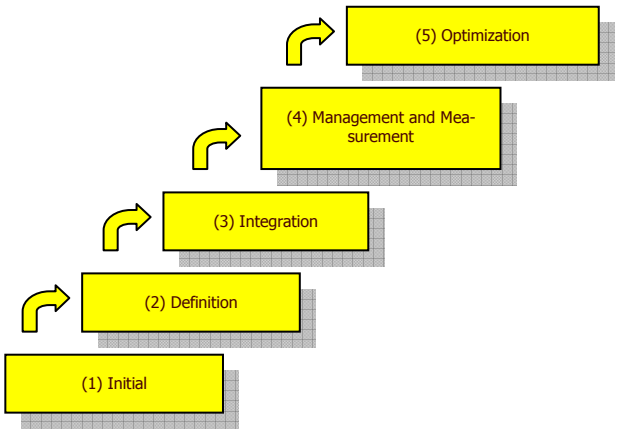

Figura 6. Schema del modello di maturità del testing (TMM).

Anche il modello TMM prevede cinque livelli di maturità del processo di collaudo. Ecco, a titolo di esempio, gli obiettivi e le caratteristiche dei singoli livelli.

Livello 1: Initial

A questo livello l'organizzazione non pone nessun obiettivo specifico al processo di testing. L'organizzazione di test utilizza un processo caotico, cioè non strutturato, e spesso coincidente con l'attività di debugging. Il test è condotto appena il software è completato ed ha l'obiettivo principale di dimostrare che esso funziona. Il prodotto software è rilasciato senza l'assicurazione qualità. Le risorse sono spesso inadeguate, il test viene fermato quando scade il tempo e

[10] Qui si fa riferimento al modello elaborato da Ilene Burnstein, Ariya Homyen, Tartip Suwanassart, Gary Safena e Rob Grom dell'Illinois Institute of Technology.

non quando abbia raggiunto gli obiettivi. Si utilizzano pochi strumenti a supporto.

Livello 2: Definition

A questo livello di maturità il testing viene distinto dal debugging ed è definito come una fase che segue la codifica e non integrata nel ciclo di vita del software. L'organizzazione inizia a pianificare il testing ma solo dopo il coding. La percezione è che il testng è basato solo sull'esecuzione del codice e quindi è pianificato dopo che il codice è stato completato. L'obiettivo principale del testing a questo livello di maturità è quello di mostrare che il codice è aderente alle specifiche. Si introducono metodi e tecniche di base del testing. I problemi legati alla qualità del software si riscontrano a causa della tardiva pianificazione dei test nel ciclo di sviluppo. Inoltre molti difetti si propagano nel codice dalle fasi di analisi dei requisiti e di disegno per l'assenza di revisioni tecniche e ispezioni.

Livello 3: Integration

A questo livello di maturità l'organizzazione fa un grande passo in avanti. Il testing non è più una fase che segue la codifica ma una perfettamente integrata nel ciclo di vita del software. L'organizzazione crea un gruppo di testing con le competenze necessarie. Il testing inizia già nella fase di analisi dei requisiti e continua durante tutto il ciclo di sviluppo secondo il modello a "V". Gli obiettivi del testing sono definiti in base ai requisiti tenendo conto delle esigenze del cliente e degli utenti, e sono utilizzati per progettare i casi di test ed i criteri di completamento. La costituzione del gruppo di test è il riconoscimento della professionalità del ruolo. La formazione tiene conto delle esigenze del testing. Si introduce l'uso di strumenti base a supporto delle attività di testing. Pur iniziando a riconoscere l'utilità del controllo della qualità, non si formalizza ancora un piano di revisioni e queste non sono condotte lungo l'intero ciclo di vita. Non è previsto ancora un programma di misurazioni per valutare la qualità del prodotto e l'efficacia del processo.

Livello 4: Management and Measurement

Il testig è ora un processo misurato e quantificato. Le revisioni tecniche e le ispezioni sono eseguite in tutte le fasi del ciclo di sviluppo e considerate come attività di controllo della qualità ed appartenenti al processo di test. Il software è testato anche nelle sue caratteristiche come l'affidabilità, l'usabilità e la manutenibilità. I casi di test provenienti dai vari progetti sono collezionati e conservati in una base dati per il riuso ed i test di regressione. I difetti sono registrati e a loro è assegnato un livello di severità. I difetti maggiori del processo di test sono ora la mancanza di un approccio alla prevenzione dei difetti ed alcune carenze nella raccolta, analisi e disseminazione delle metriche relative al test.

Livello 5: Optimization, Defect Prevention, and Quality Control

A questo livello di maturità l'organizzazione è ora in grado di misurare i costi e l'efficacia del processo di test e di ottimizzarlo. Si mette in moto un meccanismo di monitoraggio e di miglioramento continuo. Si effettua la prevenzione dei difetti ed il controllo della qualità. Il processo di testing è gestito con un approccio statistico, con misurazioni del livello di confidenza e di affidabilità. La selezione e la valutazione dei tool è fatta tramite una procedura definita. Si utilizzano strumenti automatici per la progettazione, l'esecuzione e la riesecuzione dei casi di test, la registrazione e l'analisi dei difetti, la raccolta, analisi e l'utilizzo delle metriche.

2.5.3 Attributi di un processo di test maturo

Gli attributi di un processo maturo per il testing sono descritti qui di seguito; ciascuno di essi supporta uno o più criteri di maturità descritti nel modello CMM. I ricercatori pensano che un processo maturo per il testing abbia:

Politica per il testing. Ci deve essere una politica per il testing ben definita, documentata ed applicata in tutta l'organizzazione. Tale politica sarà supportata dal management, istituzionalizzata e integrata nella cultura aziendale.

Processo di pianificazione. Ci deve essere un processo per la pianificazione del testing che sia ben definito e documentato, utilizzato in tutti i progetti e che permetta di definire gli obiettivi del test ed i valori da raggiungere, le risorse necessarie ai test, la progettazione dei casi di test, la pianificazione delle attività di test, le attività di test ed i costi relativi. I piani di test devono riflettere i rischi individuati ed i tempi e le risorse previste sufficienti al completamento delle attività.

Ciclo di vita del test. Un ciclo di vita del test ben definito contiene le fasi e le attività previste per il testing ed è perfettamente integrato con il ciclo di vita del software. Il processo di testing è molto ampio e contiene più di quanto non sia previsto dalla sola esecuzione dei test: pianificazione dei test, revisione del piano di test, progettazione del test, sviluppo del software necessario al test, aggiornamento degli output del processo di test. Esso è applicato a tutti i progetti.

Gruppo di test. E' bene che ci sia un gruppo di test indipendente dal gruppo di sviluppo[11]. La costituzione di un gruppo di test è sponsorizzata dal

[11] L'importanza di un gruppo di test indipendente dallo sviluppo è ormai un concetto acquisito nel mondo dello sviluppo software. In una piccola o media organizzazione, dove spesso non è possibile mantenere un gruppo di test permanente, il management deve garantire tale indipendenza in ciascun progetto evitando che il capo progetto possa "comandare" alle persone che svolgono il test cosa, come e quando fare. Sarà la competenza delle persone che svolgono i test a concordare in fase di pianificazione del progetto quali test eseguire, con quali obiettivi, con che tempi e con quali risorse. I risultati dei test saranno poi valutati in fase di controllo dello stato di avanzamento del progetto ed eventuali decisioni saranno prese di conseguenza dal management /dalla direzione).

management che definisce anche un piano di formazione per l'acquisizione delle competenze necessarie e mantiene il gruppo sempre motivato.

Miglioramento dei processi. Il miglioramento dei processi, incluso quello di test, deve essere sempre presente in un'organizzazione software. L'attività può essere assegnata ad un gruppo specifico dedicato a tale attività (generalmente nelle grandi organizzazioni), alla funzione di assicurazione qualità oppure allo stesso gruppo di test. Essendo il processo di test, come tutti gli altri processi, definito e misurato, la funzione deputata al suo miglioramento eserciterà la leadership sul processo misurandone le prestazioni, valutandolo rispetto agli obiettivi, proponendone i miglioramenti e valutandone gli impatti sull'organizzazione.

Metriche. Nell'ambito del programma di misurazioni esistenti nell'organizzazione[12] si definisce un set di misure da effettuare sul processo di test adoperato nei progetti, si raccolgono i dati e si valutano i risultati. I risultati della valutazione sono utilizzati per il miglioramento del processo di test.

Strumenti. Strumenti appropriate sono resi disponibili al gruppo di test per assisterli nello svolgimento dei compiti, la raccolta e l'analisi dei dati relativi al processo di test. La funzione deputata al miglioramento del processo di test identificherà e promuoverà l'utilizzo degli strumenti più adatti integrandoli nell'ambiente organizzativo.

Controllo dell'efficacia del processo. Il management monitorizza e controlla il processo di test per registrarne i miglioramenti, intraprendere le dovute azioni in caso di problemi, valutare le prestazioni e le potenzialità. L'analisi del processo per determinare l'efficacia e le potenzialità è condotta con tecniche quantitative.

Controllo della qualità del prodotto. Il raggiungimento della qualità è verificato tramite metodi statistici, mentre i criteri per valutare se "fermare il test" sono di tipo quantitativo. La qualità del prodotto è quindi monitorata, i difetti sono registrati, l'analisi causale è condotta per prevenire i difetti.

[12] Si presuppone che un'organizzazione matura abbia un programma di misurazioni per valutare le proprie performance in termini non solo di obiettivi aziendali ed obiettivi di progetto raggiunti, ma anche in termini di efficacia dei processi adoperati per la produzione ed il controllo del software.

La fase di validazione nel ciclo di sviluppo del software ricopre un ruolo di estrema importanza per la qualità del prodotto finale. Essa, infatti, permette di valutare il livello di qualità raggiunto dall'applicazione sviluppata evidenziandone gli errori e permettendone la loro correzione.

In particolare, il testing ha il compito di verificare l'aderenza ai requisiti e la correttezza dell'implementazione. In altri termini, essa verifica, da un lato, l'aderenza ai requisiti così come definiti nelle specifiche tecniche e nel piano di qualità (requisiti funzionali e requisiti qualitativi) e, d'altro, la correttezza della progettazione e della relativa implementazione.

Il presente documento fa parte di un manuale che descrive una metodologia di test completa, che indirizza sia i test statici (revisioni tecniche, ispezioni e walkthrough), sia dinamica (testing vero e proprio). La descrizione delle fasi di test include la pianificazione, la progettazione dei test e la predisposizione degli ambienti, l'esecuzione ed il controllo e monitoraggio. I livelli di test previsti sono quelli tradizionali (unitario, di integrazione, di sistema, collaudo utente). E' fatto un accenno anche ai test delle applicazioni destinate al Web (Il tema dei test delle applicazioni di e-business è trattato in dettaglio in un altro documento dello stesso autore). Sono inoltre descritte le attività di gestione delle anomalie e di gestione della configurazione di test. Il documento riporta anche le metriche relative ai test ed i metodi più comuni (white-box e black-box, top-down e bottom-up, controllo statistico della rimozione degli errori, curva di saturazione degli errori rimossi, matrice OTR ecc.).

Ercole Colonese svolge la sua attività di consulente essenzialmente nell'area delle metodologie per lo sviluppo del softwaree dei sistemi qualità. La sua esperienza è maturata in moltissimi anni di lavoro presso i laboratori internazionali IBM e organizzazioni AMS dove ha ricoperto ruoli tecnici, manageriali e dirigenziali. Ha realizzato numerosi sistemi qualità aziendali certificati ISO9000 presso piccole, medie e grandi aziende, pubbliche e private. Ha implementato modelli di eccellenza (EFQM, Malcom Baldridge, MDQ). Ha condotto diversi progetti di reingegnerizzazione dei processi di sviluppo software in ottica di miglioramento delle performance e della qualità. Ha applicato con successo i modelli di maturità dei processi SEI-CMM e CMMI. Come docente, tiene corsi e seminari sulla qualità del software e le metodologie di sviluppo presso aziende ed università.

e-mail: ercole@ colonese.it www.colonese.it