

## Quando gli errori residui nel software sono un problema ...

### Controllo qualità al servizio del business

Di Ercole Colonese (\*)

(\*) consulente, socio Aicq-ci, membro del Sottocomitato Qualità del software

Quando lavoravo in una delle maggiori aziende di software, ero responsabile dei test per una linea di prodotti diffusi in tutto il mondo. Il prodotto principale girava su Mainframe IBM mentre gli altri, detti Agent, giravano su quasi tutte le piattaforme disponibili all'epoca. I software erano apprezzati dai clienti per la loro capacità di gestire reti molto estese di distribuzione delle informazioni (Internet non era diffusa), ma lo erano ancora di più dalla compagnia per il profitto che generavano.

La qualità del software, intesa come capacità di risolvere adeguatamente le problematiche applicative che indirizzava - funzionalità e prestazioni - era costantemente discussa con i clienti che richiedevano nuovi requisiti e prospettavano nuovi scenari di utilizzo.

L'evoluzione costante delle tecnologie e la comparsa sul mercato di nuove piattaforme veniva indirizzata con nuovi rilasci. Il prodotto - meglio, la famiglia di prodotti - era in costante evoluzione, sia dal punto di vista funzionale che delle prestazioni e tecnologico. Una vera gioia per l'intera organizzazione, o quasi: gli analisti di prodotto (si chiamavano Planner) viaggiavano per tutto il mondo a raccogliere le richieste dei clienti, i progettisti disegnavano nuove architetture, gli sviluppatori sperimentavano nuove tecnologie ed i tester gioivano a "fare le pulci agli sviluppatori". Unico ad essere in apprensione, all'epoca, ero io. Perché?

C'era un aspetto particolare della qualità che era motivo di allarme per i clienti, di attenzione per il management aziendale e di ansia per me che, nel frattempo, avevo assunto anche la responsabilità della qualità dei prodotti. La caratteristica di cui parlo è la *difettosità residua del software*. Gli sviluppatori la consideravano un semplice indice di qualità, una di quelle metriche inventate da chi ha poco da fare! I clienti, al contrario, la percepivano molto negativamente: un allarme. Per loro rappresentava il numero di volte che gli utenti erano costretti ad interrompere il proprio lavoro, uno stop per il business. "Uno stupido errore nel codice che blocca il business. Assurdo, inaccettabile! Non poteva essere trovato prima che il prodotto fosse rilasciato sul mercato?" Bella domanda. E' proprio questo il punto in questione.

Ad ogni fermo dell'operatività partiva un allarme, la notifica del problema e la richiesta di una risoluzione rapida; immediata quando il problema era grave, e lo era quasi sempre, per i clienti. E se il problema non era risolto in tempi brevi, si innescava un processo di "escalation" che poteva giungere fino al quartiere generale della compagnia. Allora sarebbe stato molto difficile, oltre che imbarazzante, spiegare perché non si era stati capaci di risolvere il problema con piena soddisfazione del cliente!

A complicare lo scenario contribuiva la dislocazione dei clienti nei vari paesi del globo, ognuno operativo nel proprio fuso orario. Gli allarmi, poi, erano lanciati nelle proprie lingue (e non conoscendole non si poteva capire, per nostra fortuna, quali fossero gli impropri che accompagnavano le richieste di soccorso!).

La problematica era comunque elegantemente risolta con una struttura di supporto "in loco" che agiva presso quasi tutti i paesi in cui i prodotti erano commercializzati. La soluzione, gradita dai clienti, lo era meno per il management aziendale che vedeva così ridursi impietosamente i margini di profitto. Conclusione: la politica per la qualità doveva necessariamente puntare a ridurre drasticamente il numero di errori residui nel software rilasciato sul mercato! Occorreva però anche stimare tale numero, sulla base del quale si sarebbe potuto pianificare il servizio di supporto in loco, controllare il budget ad esso dedicato e, quindi, ridurre i costi.

All'organizzazione di test era affidata la responsabilità di verificare la qualità dei prodotti - funzioni, prestazioni e difettosità -. Alla funzione qualità era richiesto invece di "certificare" la qualità finale del prodotto e di fornire quel richiestissimo numero, l'Indice di difettosità residua (ricordo ancora un  $QI_R=0,0012$  Errori/Kloc!). Il numero magico giungeva così sul tavolo della direzione prodotti che pianificava il budget da assegnare ai gruppi di supporto dislocati nei vari paesi. Sbagliare il calcolo dell'indice era un rischio reale: un errore per difetto comportava sottodimensionare il supporto fornendo un servizio insoddisfacente per i clienti che avrebbero subito contestato; un errore in senso contrario, invece, portava a sovradimensionare i gruppi erodendo buona parte del profitto. Ora capite il motivo della mia apprensione!

La soluzione adottata consisteva nell'utilizzare tre metodiche di base:

1. "Teoria della propagazione degli errori nel software";
2. "Profilo della rimozione degli errori durante il ciclo di sviluppo";
3. "Curva di saturazione dei difetti rimossi dal test".

Era questa l'azione di contenimento del rischio che si è dimostrata davvero efficace. Ed è di queste tre tecniche che si vuole parlare nell'articolo. I temi non sono approfonditi per evidenti ragioni di spazio; sono invece delinea-

ate le caratteristiche principali che dovrebbero aiutare a valutare l'efficacia dei metodi.

L'adozione di ogni tecnica ha un suo costo. Sarà la necessità di raggiungere o meno un obiettivo di qualità così specifico a determinare se il costo sia proporzionato al beneficio. Come dire, "il fine giustifica i mezzi". E nel nostro caso lo giustificava. A garantire una maggiore efficacia dei metodi adottati contribuiva la funzione Controllo qualità, capace di verificare e valutare l'efficacia dei metodi durante l'intero ciclo di sviluppo. Quel periodo è stato infatti una grande palestra per la mia formazione.

### 1. Propagazione degli errori nel software

La teoria della propagazione degli errori nel software è stata presentata per la prima volta nel 1981 dall'IBM System Scientific Institute a fronte di una ricerca condotta sullo sviluppo del software nei maggiori laboratori della compagnia. La teoria è sintetizzata graficamente nella figura che segue.



Figura 1. Teoria della propagazione degli errori nel software.

In ogni fase del ciclo di sviluppo, gli errori provenienti dalla fase precedente sono ereditati nella fase attuale aggiungendosi a quelli immessi ex novo. Parte degli errori ereditati dalla fase precedente generano, a loro volta, ulteriori errori secondo un fattore di amplificazione 1:x. Questo fattore dipende dalla fase del ciclo di sviluppo, dalla complessità del software e dalla tipologia di errore. L'esperienza ha insegnato, ad esempio, che alcuni errori insiti nell'interpretazione dei requisiti hanno un fattore di amplificazione fino a 1:5 e si propagano in tutte le fasi successive. Così un solo errore nell'interpretazione di alcuni requisiti chiave generava fino a 5 errori nelle specifiche, fino a 25 errori nel disegno e fino a 125 errori nel codice! Alcuni errori di progettazione, invece, potevano avere un fattore di amplificazione anche 1:7 (esempio: errore nella progettazione della base dati). Solo la registrazione degli errori rilevati e l'analisi delle loro caratteristiche può farci capire quali errori siano più dannosi e come evitarli. Il modello rap-

presentato nella Figura 1 mostra quanto sia importante ridurre il numero totale di errori trasmessi alla fase successiva per ridurre, di conseguenza, quello degli errori ricevuti in eredità da quella precedente. A tale scopo si deve agire su due fronti: ridurre il numero di errori generati ex novo in ogni fase e aumentare il numero di errori rimossi in ogni fase.

L'attività di "revisione tecnica" (ispezione) eseguita nelle fasi alte del processo di sviluppo è cruciale per migliorare la qualità del software e ridurre i costi relativi alla correzione degli errori rilevati durante il test e l'esercizio.

L'utilizzo di tabelle con la classificazione ortogonale dei difetti<sup>1</sup> e la contemporanea attività di analisi degli errori più frequenti (Pareto) completa l'attività di controllo della qualità e dei costi. Dati statistici su progetti reali dimostrano una riduzione dei costi complessivi di rimozione dei difetti del 75% con l'applicazione di tali metodiche.

### 2. Profilo di difettosità della rimozione degli errori

Abbiamo visto come la teoria della propagazione degli errori nel software agisca in maniera subdola ed impietosa.

La tecnica, detta "Profilo di rimozione degli errori nel software", è stato di grande aiuto a gestire il problema.

Si tratta di pianificare il progetto di sviluppo prevedendo già una capacità di rimozione degli errori in ogni fase.

In Tabella 1 la riga superiore ("Piano") mostra un esempio dei valori attesi per tale capacità di rimozione, mentre la riga sottostante ("Risultati") riporta i valori conseguiti dalle attività di rimozione.

Tasso di rimozione	REQUISITI	SPECIFICHE TECNICHE	DISEGNO	CODIFICA & TEST UNITARIO	TEST D'INTEGRAZIONE	TEST DI SISTEMA
Piano	3,9	9,2	13,8	21,4	12,0	0,12
Risultati	1,2	3,7	11,2	27,4	13,4	5,2

Tabella 1. Profilo di rimozione degli errori in un progetto software.

Nell'esempio riportato, le fasi alte del ciclo sono caratterizzate da una rimozione effettiva inferiore a quella attesa; tale situazione porta la difettosità intrinseca del software ad essere maggiore di quella prevista e sarà evi-

<sup>1</sup> Si tratta della tecnica conosciuta come "Orthogonal Defect Classification", ODC, che permette di valutare l'efficacia e l'efficienza della rimozione degli errori e di costruire il profilo di immissione degli errori.

denziata con numero maggiore di errori rilevati durante le fasi di test e in esercizio. I valori a consuntivo rilevati in tali fasi confermano, purtroppo, la previsione. Gli utenti sperimenteranno un software di qualità inferiore alle attese.

La metrica utilizzata misura il numero di errori rimossi per unità di prodotto (KLoc oppure FP) in ogni singola fase. Si tratta quindi di una misura normalizzata. Un software di 10.000 FP, per esempio, che venga messo in esercizio con 50 errori presunti avrà un "Indice di difettosità residua":

$$QI = 50/10.000 = 0,005 \text{ Errori / FP.}$$

Ma come definire i valori giusti in fase di pianificazione? Con "l'esperienza e l'analisi dei dati statistici".

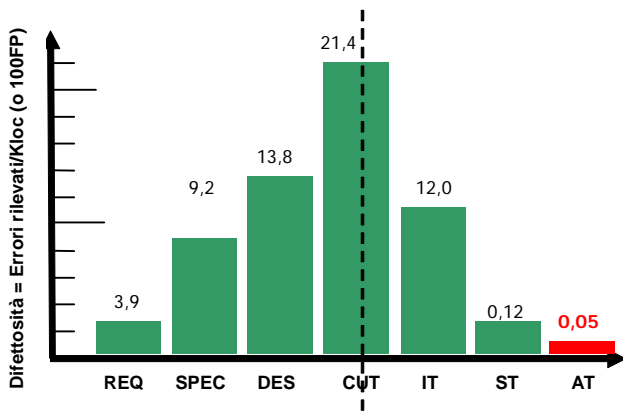
E se non fossero disponibili – come purtroppo accade nella maggior parte dei casi - dati statistici? La risposta spontanea sarebbe quella in voga presso i media:

*"No dati, no party!"*.

Ma qualcosa possiamo e dobbiamo fare, comunque.

E' sufficiente, per esempio, iniziare a registrare il numero di errori rimossi in ogni fase tramite le ispezioni nelle fasi alte del ciclo e con i test quando il codice è pronto.

La rappresentazione in istogramma dei valori registrati permette di costruire un profilo come quello mostrato nella Figura 2.



**Figura 2.** Profilo di rimozione degli errori nel software.

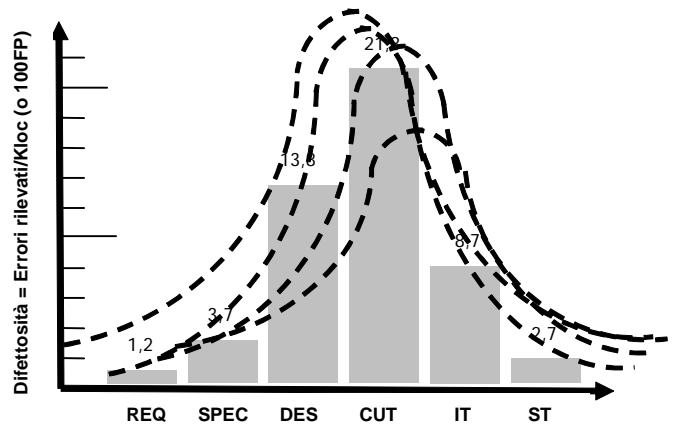
La linea tratteggiata verticale mostra il confine tra la rimozione eseguita con le ispezioni e quella con i test.

Il codice è sottoposto ad entrambe le tecniche di rimozione: le ispezioni sulle parti di codice più critiche ed il test unitario a copertura di tutto il software.

Il numero di errori rimossi dipende da molti fattori. I più importanti sono riassumibili in: complessità del progetto, tecnologia utilizzata, processo di sviluppo adoperato, competenza delle persone. Ciò significa che in progetti simili per complessità e tecnologie, un gruppo di lavoro

che adotti lo stesso processo ed abbia pari competenza sarà in grado di rimuovere mediamente lo stesso numero di errori.

Ma come risulterebbero le curve per altri progetti realizzati da differenti gruppi di lavoro? La Figura 3 ne mostra un esempio significativo.



**Figura 3.** Curve di Raleigh relative a progetti differenti.

Lo studio eseguito su moltissimi progetti diversi per tipologia e dimensioni, sviluppati in diversi contesti culturali e con differenti processi mostra l'andamento delle curve riportate nella figura. L'insieme di curve mostrate sono conosciute come "curve di Raleigh".

La forma a campana delle curve rimane costante mentre varia l'altezza e l'ampiezza rispetto alle fasi del ciclo. La ricerca ha evidenziato che curve spostate verso sinistra hanno un andamento più dolce nella parte destra. Il risultato pratico, di grande rilievo per il nostro mestiere, è che aumentando la rimozione degli errori nelle fasi alte del ciclo si riduce di conseguenza il numero di errori rilevati durante i test e, ancora più importante, in esercizio.

L'analisi statistica condotta sui progetti realizzati ci ha permesso di valutare la difettosità residua nel primo periodo di esercizio (i primi 6 mesi) pari a metà di quello rilevato nell'ultima fase di collaudo (test di sistema o collaudo utente). Il rimanente 50% dei difetti è rilevato nei successivi 12 mesi di esercizio.

Il profilo di rimozione degli errori ci fornisce quindi una prima ricetta da seguire: "rimuovere quanti più errori nelle fasi alte del ciclo procura benefici nelle fasi successive". Forse lo sapevamo già, ma spesso ce ne dimentichiamo. Vediamo ora come confrontare i risultati ottenuti con quelli attesi.

### Valori a consuntivo paragonabili a quelli attesi

Quando i valori effettivi sono paragonabili a quelli previsti (anche se non con precisione assoluta) possiamo confermare il modello statistico utilizzato e la bontà dei dati presenti nell'archivio storico dei progetti. I risultati otte-

nuti sono convalidati e l'indice di difettosità residua del software sarà quello previsto.

#### Valori a consuntivo inferiori alle attese

In questo caso si ipotizzano due situazioni:

A) Il software realizzato è qualitativamente superiore a quello previsto. Il caso potrebbe essere determinato da una maggiore competenza delle persone e/o da una minore complessità del lavoro rispetto a quanto stimato. Occorre quindi analizzare in dettaglio i dati per confermare o meno l'ipotesi. La maggiore efficacia ed efficienza dell'organizzazione potrebbe essere stata indotta da azioni di miglioramento intraprese. Occorre capire, allora, perché non sia stata presa in considerazione in fase di pianificazione. Si esclude invece un miglioramento casuale e spontaneo, non generato da alcuna azione! L'analisi potrebbe evidenziare anche una minore complessità del progetto rispetto a quella stimata e quindi giustificare la maggiore qualità del software. La registrazione dei dati nell'archivio storico dei progetti permette di aggiornare i modelli statistici adoperati. I prossimi progetti potranno usufruire dei dati aggiornati ed essere stimati tenendo conto dei miglioramenti conseguiti.

B) Le attività di revisione tecnica e di test sono poco efficaci. Gli elementi che contribuiscono a tale risultato possono essere l'inadeguata competenza o impegno delle persone che hanno svolto le attività e/o la maggiore complessità del lavoro rispetto a quella stimata. Anche in questo caso occorre analizzare in dettaglio i dati e, in base ai risultati, intervenire opportunamente (esempio: sulla competenza, l'efficacia e la motivazione delle persone, sui metodi di conduzione delle revisioni tecniche, sui metodi di progettazione dei casi di test e sul controllo dell'esecuzione dei test).

#### Valori a consuntivo superiori alle attese

In questo caso siamo in presenza di un software di qualità inferiore a quella prevista. Le attività di revisione e di test rilevano un numero di errori superiore alla media di progetti simili. La maggiore difettosità si ripercuoterà negativamente sulle fasi successive con aumento dei costi di rimozione e insoddisfazione degli utenti. L'azione più opportuna, in questo caso, è di intervenire subito nella fase in cui la deviazione è stata evidenziata. L'analisi della deviazione dai valori attesi potrà identificare le cause. E' quasi certo che si tratti di una minore qualità dei prodotti intermedi realizzati (poca cura nell'analisi dei requisiti, incompletezza nella stesura delle specifiche, pochezza della progettazione, ecc.). Le azioni di recupero devono essere immediate: eseguire di nuovo le attività di fase incriminate (per esempio, rifare parte dell'analisi dei requisiti e confrontarsi con gli utenti, rivedere la progettazione, ecc.). Occorre riportare il progetto immediatamente sui valori attesi.

### 3. Curva di saturazione degli errori rimossi dal test

La rappresentazione del numero cumulativo di errori rilevati ogni giorno durante lo svolgimento di una fase di test prende la forma mostrata nella Figura 4.

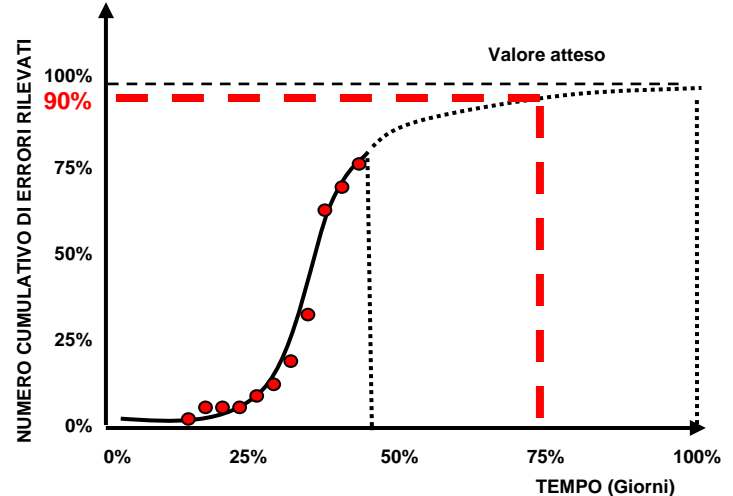


Figura 4. Curva di rimozione degli errori in una fase di test.

L'altezza della curva può variare a seconda del numero di errori rilevati; la sua forma è invece una costante del software, avvalorata dall'esperienza di numerosi progetti presi in esame.

La curva è costruita giorno per giorno riportandovi il numero di errori rilevati in forma cumulativa.

All'inizio dei test la curva stenta a salire a causa degli errori bloccanti che impediscono di esercitare gran parte del software e scoprire ulteriori errori.

Intorno al 25% del test, gli errori bloccanti sono tutti corretti ed i casi di test possono proseguire speditamente rilevando un alto numero di errori: la curva sale rapidamente!

Al completamento di circa il 50% dei test la curva sale ancora ma meno rapidamente: inizia una leggera flessione.

Arrivati al 75% del completamento dei test, la curva conferma una tendenza ad appiattirsi. L'asintoto verso cui tende la curva rappresenta il numero massimo di errori che la fase di test è in grado di rilevare al suo completamento (100%) con i casi di test progettati.

Proseguire con altri test, oltre il 100%, non porta necessariamente a trovare altri errori.

L'asintoto è calcolato su base statistica utilizzando i dati riportati nel "profilo di rimozione" per la fase in oggetto (vedi paragrafo precedente). Tale valore è il risultato della capacità di rilevare errori con casi di test progettati secondo le tecniche ed i metodi adottati in azienda per pro-

getti simili in termini di complessità, tecnologia, processo e competenza,

L'utilizzo di entrambe le tecniche ("profilo di rimozione degli errori" e "curva di saturazione degli errori") permette di controllare statisticamente l'efficacia della rimozione degli errori e di stimare il numero massimo di errori rimossi e di quelli residui.

Un modo semplice di costruire la curva di saturazione degli errori durante una fase di test consiste nel registrare ogni giorno il numero di errori rilevati e riportare il valore cumulativo su di un grafico come quello mostrato nella figura precedente.

Oltre che manualmente, come suggerito, la curva può essere costruita utilizzando uno strumento informatico in cui registrare i dati giornalieri e sui quali viene applicata la formula proposta da Musa e Ackermann e definita dalla funzione E(t):

$$E(t) = E_0 (1 - e^{-(E_0/E_{Max})t})$$

dove:

$E_0$  è il numero di errori rilevati il primo giorno di test lanciando tutti i casi di test progettati.

$E_{Max}$  è il numero totale di malfunzionamenti attesi, cioè il valore dell'asintoto verso cui deve tendere la curva.

t è il tempo di test espresso in giorni.

La formula è riportata solo per amore di completezza in quanto poco o per nulla utilizzata nella pratica.

Purtroppo, anche la tecnica manuale con interpolazione della curva è poco o mai utilizzata. Risulta ancora difficile convincere le persone - sia i tester che il capo progetto - a registrare ogni giorno gli errori rilevati. L'utilizzo di un tool per la registrazione degli errori faciliterebbe la cosa; ma credo non si tratti di un problema di "semplicità".

La tecnica ha un secondo scopo ugualmente utile ed importante come il primo: permette di calcolare il rapporto costi/benefici relativo ad un'eventuale interruzione del test ad un determinato livello di completamento.

Nell'esempio rappresentato dalla Figura 4, le linee tratteggiate, in colore rosso e più marcate, indicano che al 75% di completamento dei casi di test pianificati si è raggiunto un valore di rimozione degli errori molto interessante (90% o più). In tale caso si potrebbero concludere le attività di test potendo stimare un numero basso di errori residui. Diverso è il caso personale presentato nell'articolo, dove la rimozione degli errori andava portata fino in fondo per scoprirne il maggior numero possibili.

## Come calcolare il numero di errori residui

Il valore della difettosità residua è uguale a quello rilevato all'ultima fase di test eseguita. Questo dice l'esperienza di molti anni di controllo qualità. Nel caso rappresentato nella Figura 5 la difettosità residua del software corrisponde a 1,7 errori/KLoc (equivalente a circa 0,017 errori/FP), difettosità dell'ultima fase di test (ST).

La distribuzione degli errori nel periodo di esercizio, diviso generalmente in semestri (6 mesi, 12, mesi e 18 mesi), dipende dall'utilizzo che ne faranno gli utenti. Generalmente, per un utilizzo del prodotto a regime in condizioni normali, si prevede una distribuzione di errori pari al 50% del totale nei primi 6 mesi e del restante 50% nei successivi 12 mesi. La Figura 5 ne presenta un esempio significativo.

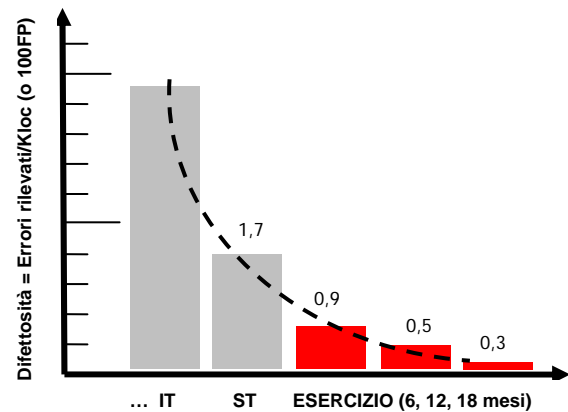


Figura 5. Difettosità residua in esercizio.

## Conclusioni

Quanto esposto si applica bene a software in cui la difettosità in esercizio sia estremamente importante.

Si evidenzia inoltre la tesi formulata nel titolo dell'articolo. Il controllo qualità non è mai fine a se stesso, come purtroppo è spesso visto o magari interpretato dai responsabili; esso è, e deve esserlo, un processo integrato nel business, un processo di supporto al business, quando interpretato nel giusto modo!

A quei tempi, al gruppo di test era devoluto anche il 30-35% del budget complessivo del progetto (bei tempi!). Un tale budget non poteva che diventare oggetto di cupidigia da parte di altre organizzazioni, per esempio, di quella di sviluppo. Mettendo in atto una propria strategia, la direzione sviluppo prodotti non perdeva occasione di enfatizzare quanto fosse spropositata la percentuale di budget dedicata al test. "Con metà del budget allocato al test ed alla qualità - fu detto un giorno durante una riunione di direzione - lo sviluppo potrebbe realizzare molto più software ad una qualità superiore!". Trattandosi di

persona intelligente, colta e di grande esperienza, si capì che si trattava di pura provocazione - almeno così interpretai io, che al tempo ero la parte contestata -. Ma la direzione generale, anch'essa preparata e di non minore esperienza, replicò seccamente: "Purtroppo lo sviluppo del software è un'attività ad alto contenuto intellettuale e quindi soggetta ad errori. Il test è quindi necessario. Finché lo sviluppo non sarà in grado di produrre software di alta qualità con le risorse attualmente allocate, non ha senso parlare di riduzione dei test o del controllo qualità". Fine della discussione che, per quanto mi ricordi, non fu più ripresa negli anni.

Personalmente penso che lo sviluppo possa produrre software di qualità migliore puntando su professionisti qualificati che conoscano bene le metodologie di sviluppo, applichino tecniche e metodi di comprovata efficacia e dominino costantemente le tecnologie, specialmente quelle nuove. Purtroppo non è facile dotarsi di gruppi di lavoro con personale qualificato in tal senso. Le aziende che hanno personale di tale livello è bene che lo mantengano con cura. Possiedono l'asset di maggiore valore per un'impresa di servizi software.

Ma non dobbiamo scoraggiarci, anche se il gap formativo è profondo e tende ad ampliarsi nel nostro paese. E' compito di ciascuno di noi perseverare nella diffusione della qualità ad ogni livello, in ogni contesto, in ogni momento!