

Aspetti di Affidabilità (Sw Reliability)

- Informalmente
 - credibilità del prodotto software
- Formalmente
 - probabilità che il prodotto software lavori “correttamente” in un determinato intervallo temporale

Difetto, Guasto, Errore

- **Difetto (defect)**
 - anomalia presente in un prodotto Sw
- **Guasto (failure)**
 - comportamento anomalo del prodotto Sw dovuto alla presenza di un difetto
- **Errore**
 - azione errata di chi (per ignoranza, distrazione, etc) introduce un difetto nel prodotto Sw

Affidabilità Sw

- Intuitivamente:
 - Un prodotto software con molti difetti è poco affidabile.
- E' chiaro che:
 - L'affidabilità del prodotto migliora via via che si riduce il numero di difetti

Caratteristiche dell'affidabilità Sw (1)

- Relazione non-semplice tra:
 - affidabilità osservata
 - e numero di difetti latenti
- L'eliminare difetti dalle parti del prodotto raramente usate
 - Ha piccoli effetti sull'affidabilità osservata.

La regola 10-90

- Esperimenti condotti su programmi di notevoli dimensioni mostrano che:
 - Il 90% del tempo di esecuzione totale è speso eseguendo il solo 10% delle istruzioni
- Detto 10% è chiamato :
 - core (nucleo) del programma

Caratteristiche dell'affidabilità Sw (2)

- Il miglioramento dell'affidabilità per l'eliminazione di un difetto:
 - dipende dalla localizzazione del difetto (ovvero se appartiene o meno al nucleo del programma)

Caratteristiche dell'affidabilità Sw (3)

- Dunque, l'affidabilità osservata dipende da:
 - come è usato il prodotto
 - in termini tecnici, dal suo profilo operativo (**operational profile**)

Caratteristiche dell'affidabilità Sw (4)

- Dunque, poiché utenti differenti usano il software secondo profili operativi diversi:
 - I difetti che si manifestano per un utente
 - potrebbero non manifestarsi per l'altro
- Dunque, l'affidabilità di un prodotto Sw:
 - Dipende dall'utente

Confronto tra affidabilità Hw e Sw

(1)

- **I guasti Sw:**
 - sono dovuti alla presenza di difetti nei programmi
 - il software non si consuma
- **I guasti Hw son quasi sempre dovuti a:**
 - consumo/deterioramento dei componenti
 - qualche componente non si comporta più come specificato
 - qualche componente si rompe

Confronto tra affidabilità Hw e Sw (2)

- Esempi di difetti Hw
 - un resistore si altera
 - un condensatore va in corto
 - una porta logica si blocca su 1 oppure 0
- Per riparare un difetto hw:
 - si sostituisce il componente

Confronto tra affidabilità Hw e Sw (3)

- I difetti Sw sono latenti
 - il sistema Sw continua a guastarsi
 - a meno che non si effettuino le dovute correzioni

Confronto tra affidabilità Hw e Sw

(4)

- A causa della differenza negli effetti dei difetti:
 - Le metriche usate per l'affidabilità Hw
 - Non sono estensibili al Sw

Confronto tra affidabilità Hw e Sw (5)

- Dopo la riparazione dell'Hw
 - la sua affidabilità torna come era
- Dopo la riparazione del Sw:
 - la sua affidabilità può aumentare o diminuire.

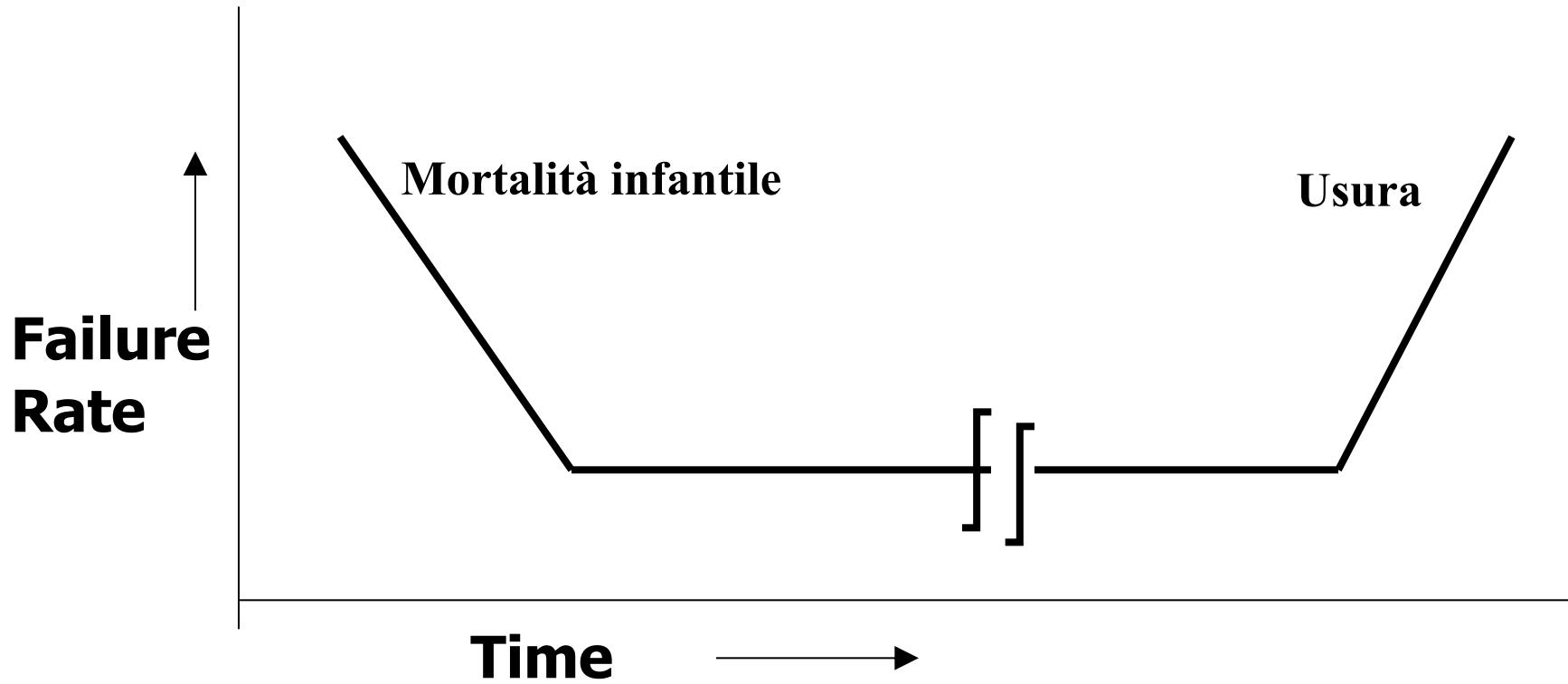
Confronto tra affidabilità Hw e Sw

(6)

- Obiettivo dell'affidabilità Hw :
 - **stabilità** (cioè tenere la frequenza di guasto costante)
- Obiettivo dell'affidabilità Sw:
 - **crescita di affidabilità** (cioè far decrescere la frequenza di guasto)

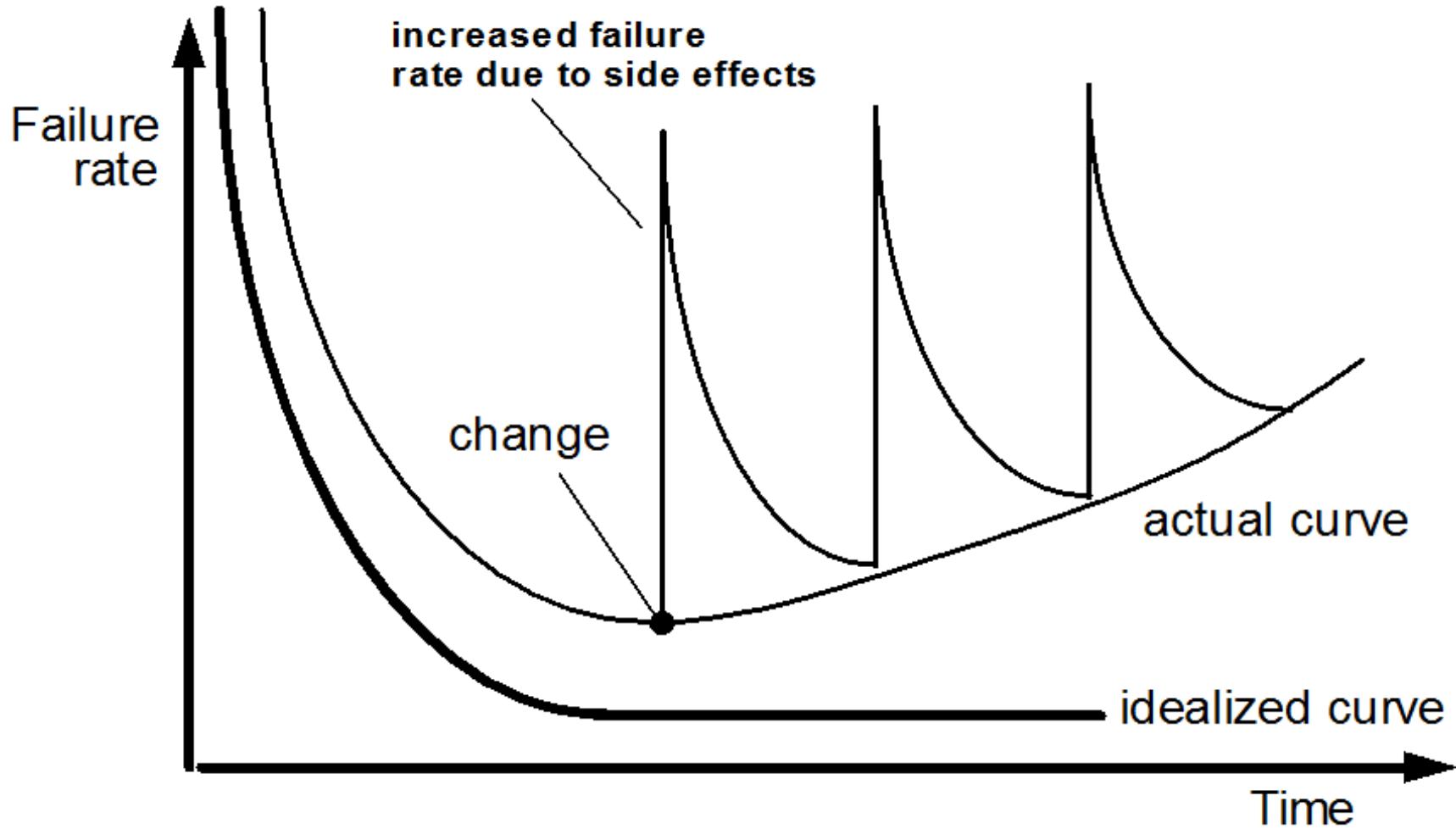
Nella realtà: andamento frequenza di guasto hardware

(effetto dell'eliminazione dei componenti difettosi prima, e dell'usura poi)



Andamento frequenza di guasto software

(effetto dell'eliminazione dei difetti prima, e
dell'invecchiamento per manutenzione poi)



Disponibilità (Sw Availability)

- % del tempo che il Sw è risultato usabile nel corso della sua vita
- Dipende
 - dal numero di guasti che si verificano
 - dal tempo necessario a ripararli

Importanza di Sw Reliability/Availability

- Metriche importanti per sistemi in cui
 - la caduta del servizio crea cadute di efficienza e sicurezza (perdite economiche e sociali)
 - sistemi di trasporto
 - di governo del traffico aereo
 - di governo del volo
 - di produzione e distribuzione di energia
 - di comunicazione
 - etc

Conclusioni (1)

- Nel corso degli anni la produzione del software ha seguito varie fasi:
 - fase **di abilità**, nella quale prevalgono gli aspetti di lavoro individuale e creativo
 - fase **artigianale**, nella quale il software viene prodotto da piccoli gruppi specializzati, spesso di alto livello di professionalità
 - fase **industriale**, nella quale l'attività di sviluppo e manutenzione del software viene pianificata e coordinata, ed il lavoro del progettista viene sempre più supportato da strumenti automatici.

Conclusioni (2)

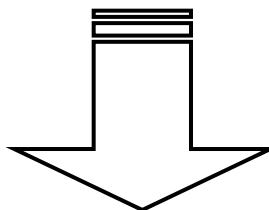
- Il termine «**ingegneria del software**» viene coniato per la prima volta nel 1968 in una conferenza NATO a Garmisch (Germania) per testimoniare l'esigenza che il software fosse inquadrato all'interno di una disciplina ingegneristica.
- Lo standard **IEEE Std. 610.12** (1990) ha formulato una definizione più completa:
 1. Applicazione di un approccio sistematico, disciplinato e misurabile allo sviluppo, esercizio e manutenzione del software, cioè applicazione di principi ingegneristici al software
 2. Studio degli approcci di cui al punto 1

Conclusioni (3)

- Il software può essere considerato come un insieme di elementi che formano una "**configurazione**" che include:
 - programmi
 - documenti
 - dati multimediali
- Viene realizzato dall'ingegnere del software applicando un **processo** che conduca a risultati di **qualità** elevata
- Come per ogni altro prodotto di successo, si applica al software un approccio ingegneristico
- Caratteristiche del software:
 - il software va "ingegnerizzato"
 - il software non si consuma
 - il software è complesso, invisibile, si conforma, si cambia

Conclusioni (4)

- Come assicurare la qualità del software che si produce?
- Come bilanciare la "domanda" crescente pur mantenendo il controllo del budget a disposizione?
- Come aggiornare applicazioni vecchie (*legacy*) ma ancora necessarie?
- Come evitare tempi di consegna più lunghi di quelli pianificati?
- Come applicare con successo le nuove tecnologie software?



I metodi e le tecniche di **Ingegneria del Software** hanno lo scopo di fornire le risposte a tali problemi, al fine di realizzare software con le desiderate caratteristiche di qualità.

I miti (da sfatare) del software

- In caso di ritardo, basta aumentare il numero di programmatore
- Una descrizione generica è sufficiente a scrivere i programmi. Eventuali modifiche si possono facilmente effettuare in seguito
- Una volta messo in opera il programma, il lavoro è finito
- Non c'è modo di valutare la qualità fino a quando non si ha a disposizione il prodotto finale
- L'ingegneria del software è costosa e rallenta la produzione