

# Software Engineering

Ercole Colonese, Senior Consultant

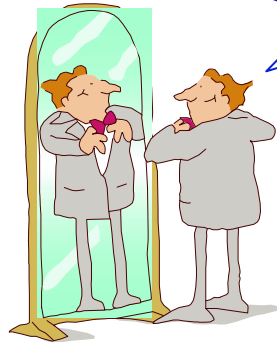
**Error injection and removal**

Rome, June 2005



# Un approccio consolidato allo sviluppo del software

*Ragazzi, iniziate pure a programmare che io vado dal cliente a sentire cosa vuole!*



**Ovvero, la rivincita di  
eXtreme Programming  
(XP)**

## Agenda

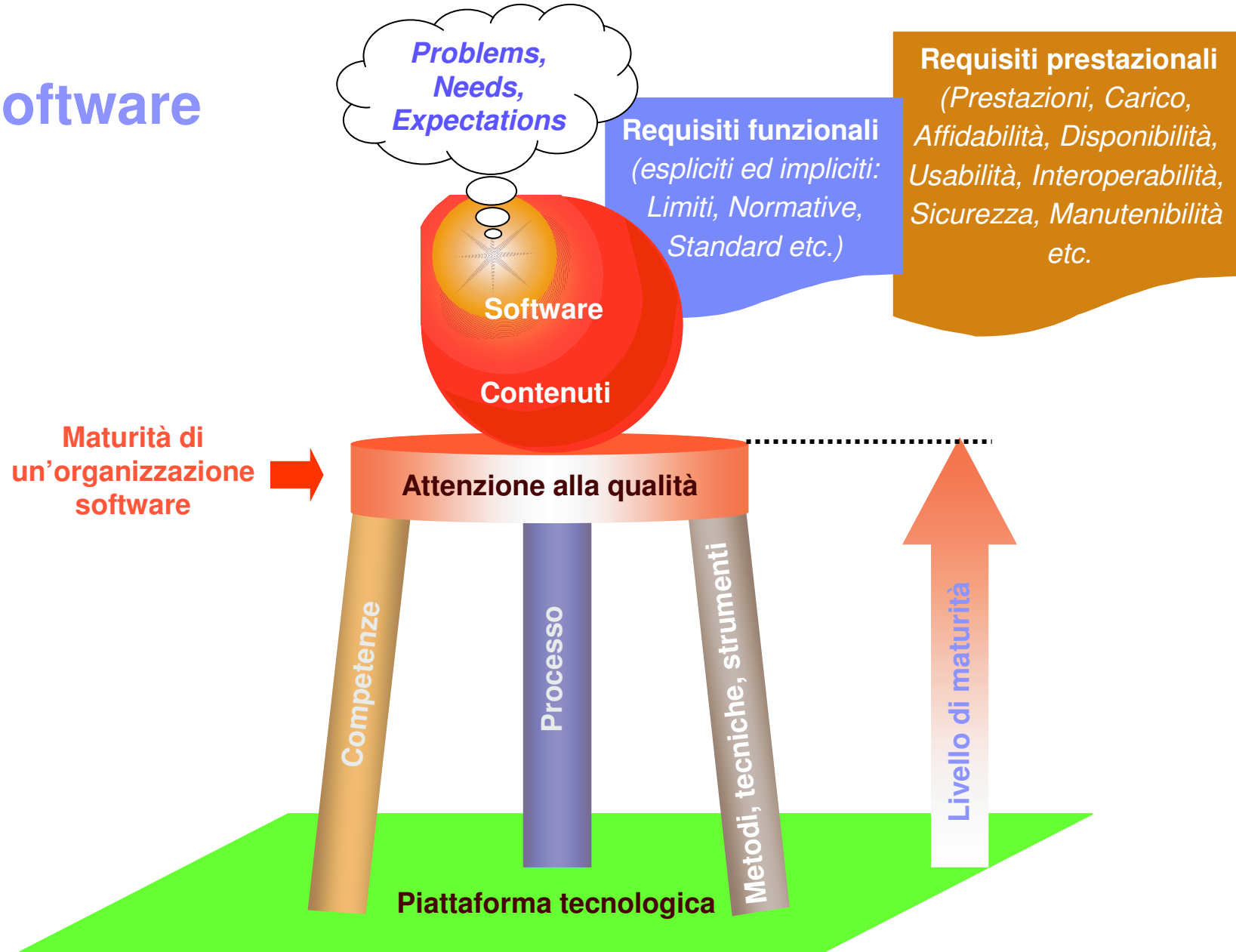
- *Alcuni elementi di riflessione sullo sviluppo del software*
- *Propagazione degli errori*
- *Curve di Raleigh*
- *Testing*
- *Bibliografia*



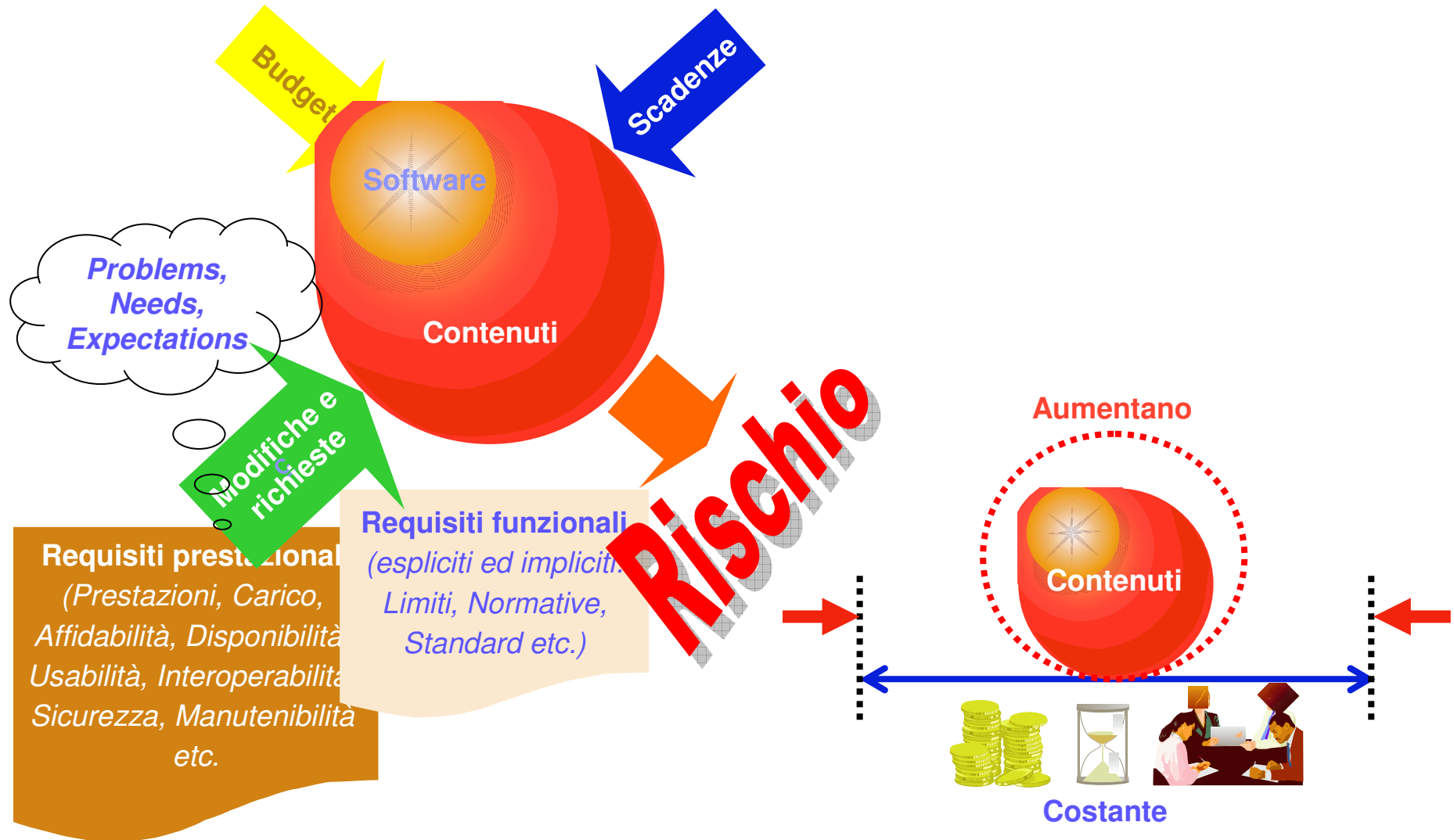
*Alcuni elementi di  
riflessione sullo sviluppo  
del software*



# Il software



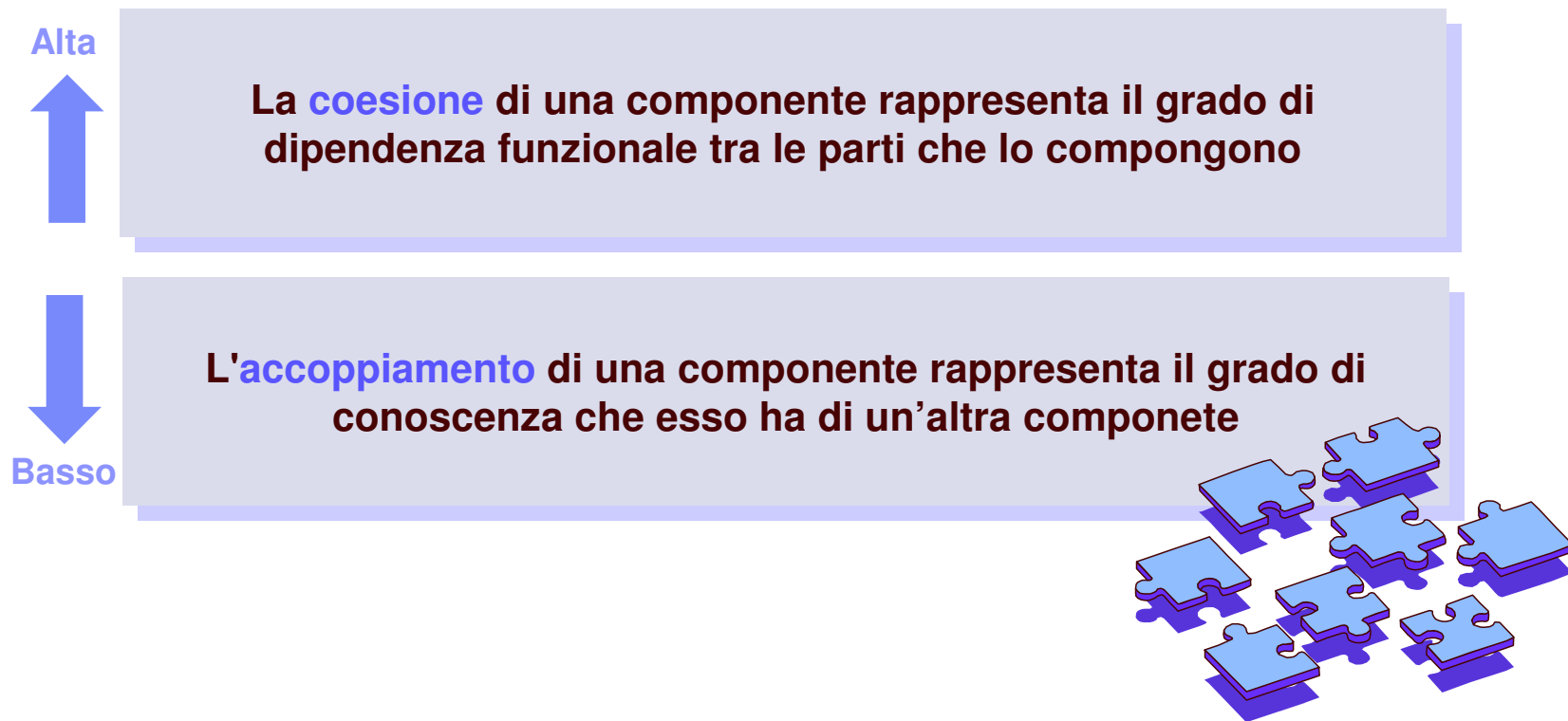
## Pressioni sui progetti software



## Qualità del software

- Modularità delle componenti ai vari livelli:

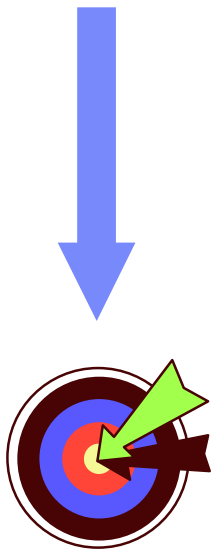
"alto livello di **coesione**" e "basso livello di **accoppiamento**"



# Coesione

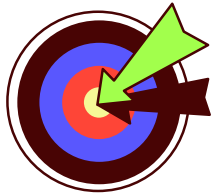
Esistono sette livelli di coesione, in ordine crescente:

1. **Casuale:** Nel modulo esistono parti distinte di codice non correlate tra di loro e senza motivi validi che lo giustifichino (esempio: funzioni diverse)
2. **Associazione logica:** Il modulo contiene del codice che realizza funzioni distinte ma correlate (esempio: aggiornamento di tabelle correlate ma di funzioni diverse)
3. **Temporale:** Il modulo contiene del codice logicamente distinto, ma da eseguire nello stesso momento (esempio: inizializzazioni)
4. **Procedurale:** Il modulo esegue funzioni non correlate tra di loro ma da eseguire in sequenza logica
5. **Comunicazione:** Le componenti del modulo, pur non correlate funzionalmente tra di loro, fanno riferimento agli stessi input e/o output
6. **Sequenziale:** Il modulo esegue più funzioni distinte che vanno eseguite in stretta sequenza (es.: l'output di una funzione costituisce l'input per la successiva)
7. **Funzionale:** Il modulo esegue una ed una sola funzione (macrofunzione)



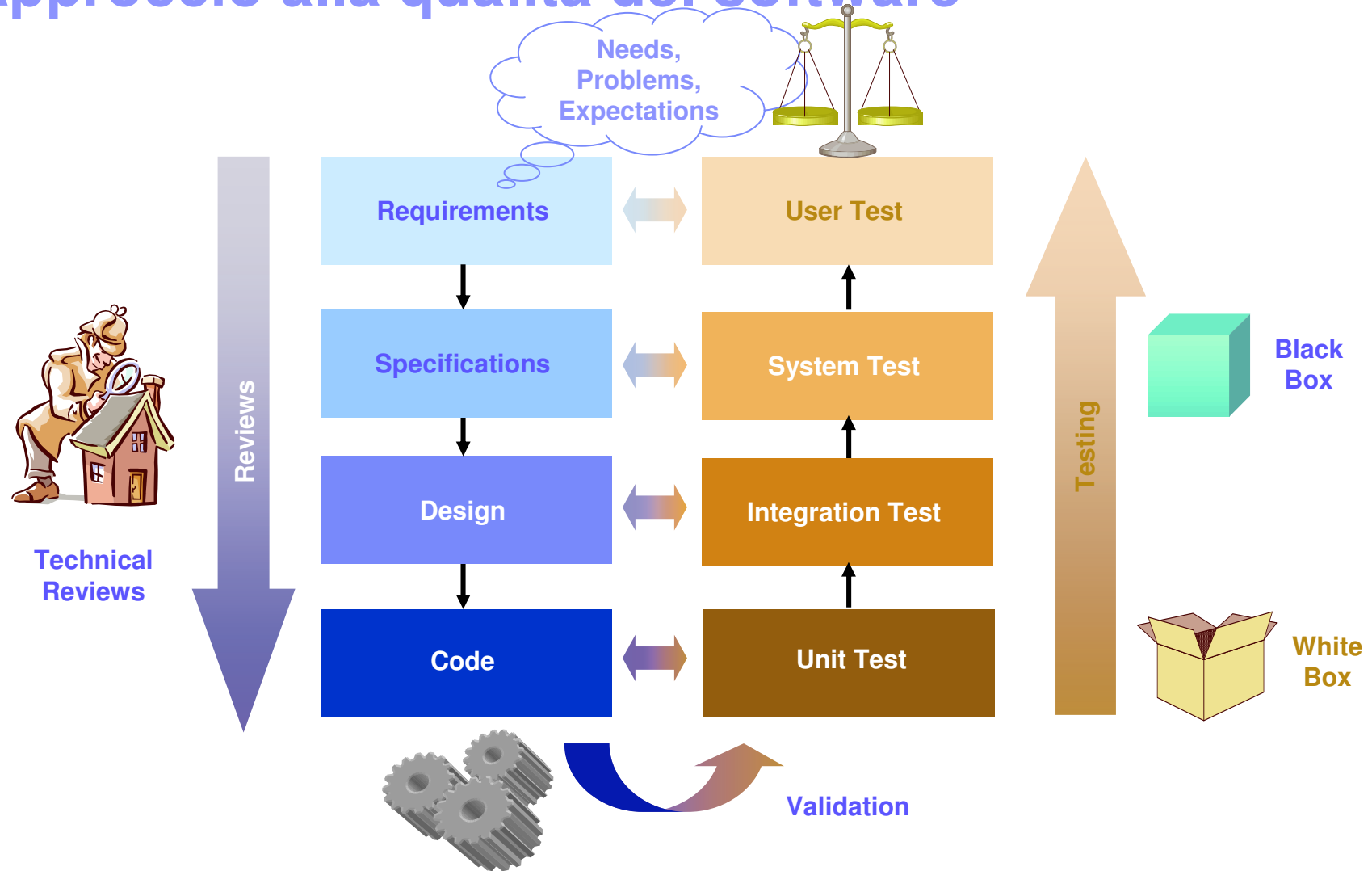
# Accoppiamento

Esistono sei livelli di accoppiamento, in ordine crescente:

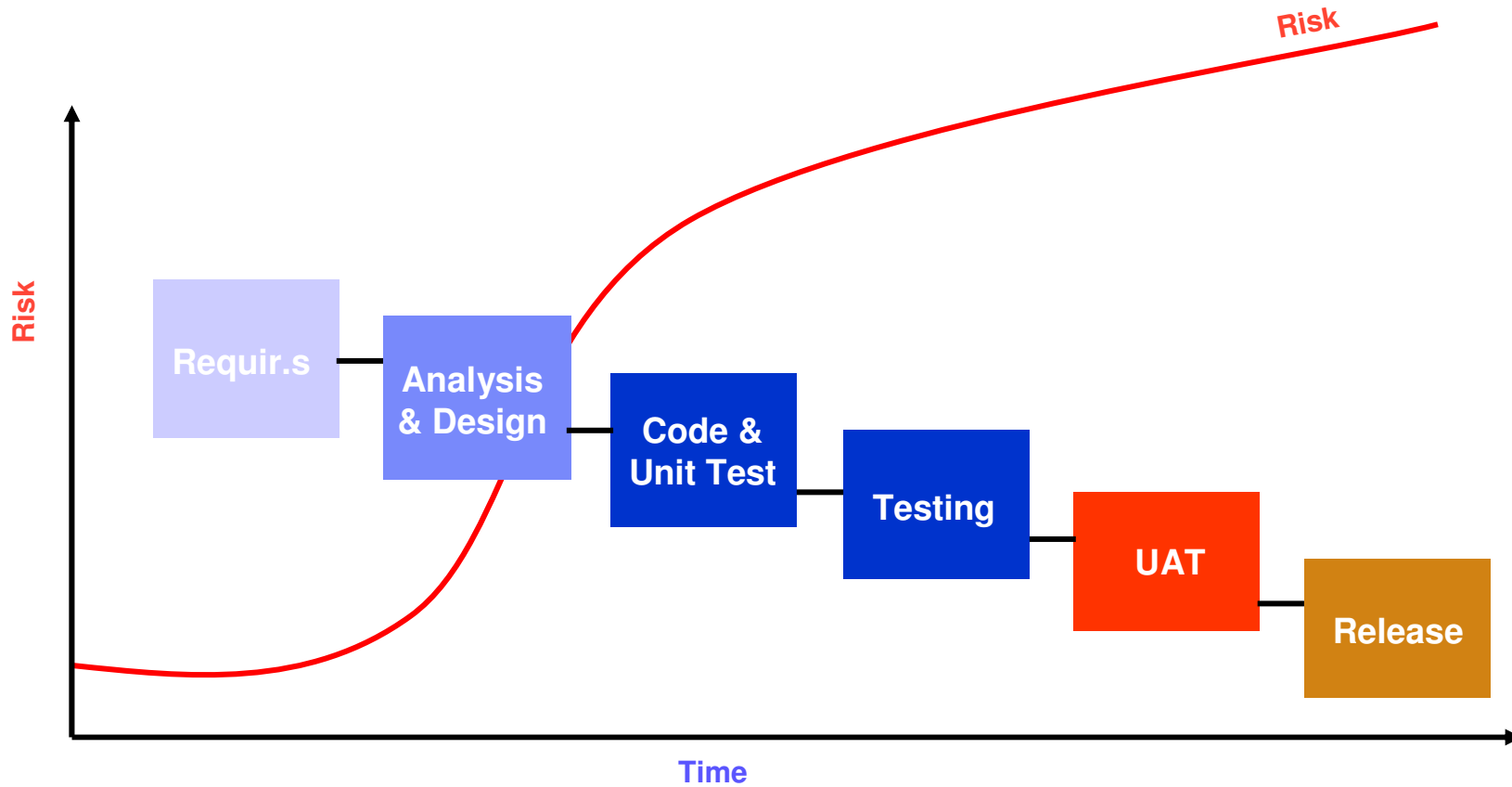


1. **Per dato:** La comunicazione tra i moduli avviene esclusivamente tramite parametri e codici di ritorno (es.: call con parametri)
2. **Per strutture dati:** La comunicazione tra i moduli avviene tramite l'accesso condiviso ad una struttura dati globale (es.: record dati)
3. **Per controllo:** La comunicazione tra i moduli avviene tramite parametri di esecuzione (es.: call con parametro "exec")
4. **Per elementi esterni:** Il modulo fa riferimento ad un simbolo definito in un altro modulo, dichiarandolo, per esempio, come "External"
5. **Per aree dati comuni:** Il modulo condivide con altri moduli l'accesso ad una intera area dati (esempio: data common area)
6. **Per contenuto:** Il modulo fa riferimento diretto alla struttura fisica di un altro modulo (esempio: salto ad una label di un altro programma)

# Approccio alla qualità del software

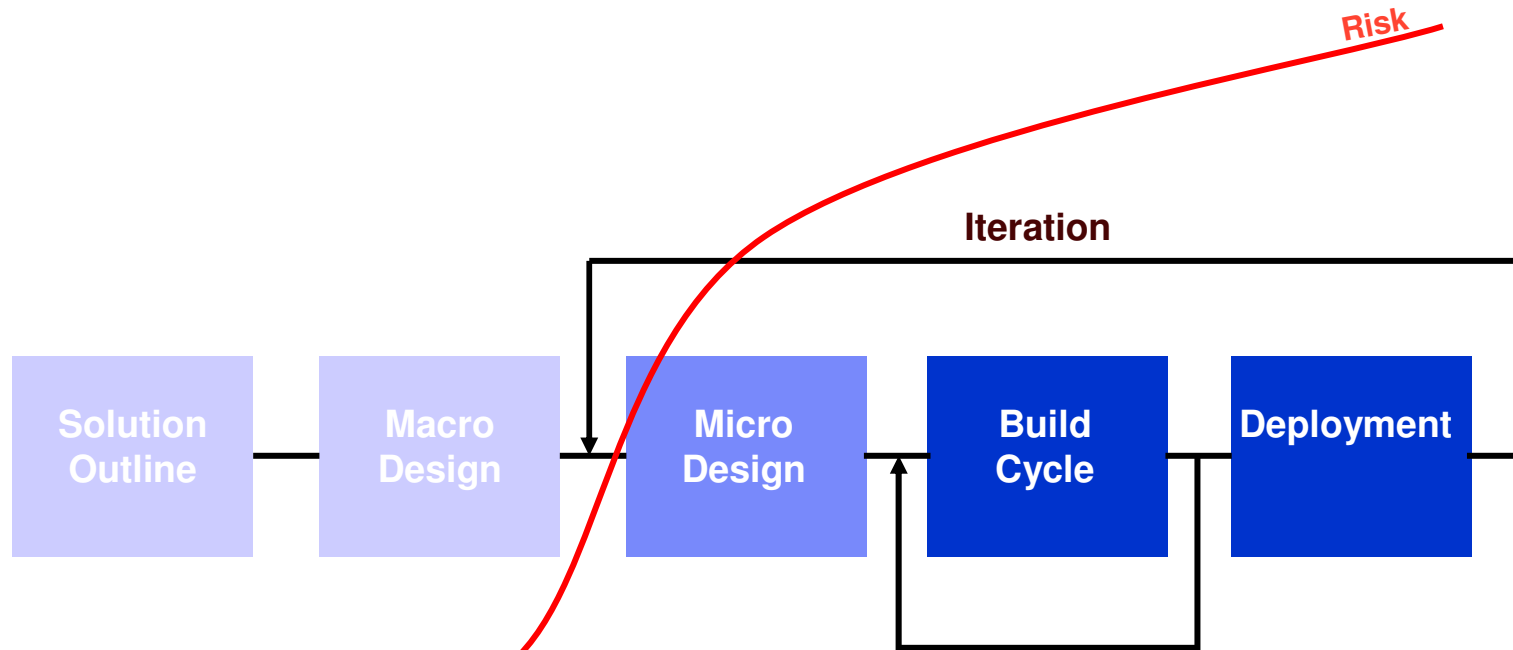


# Ciclo di sviluppo tradizionale (“a cascata”)



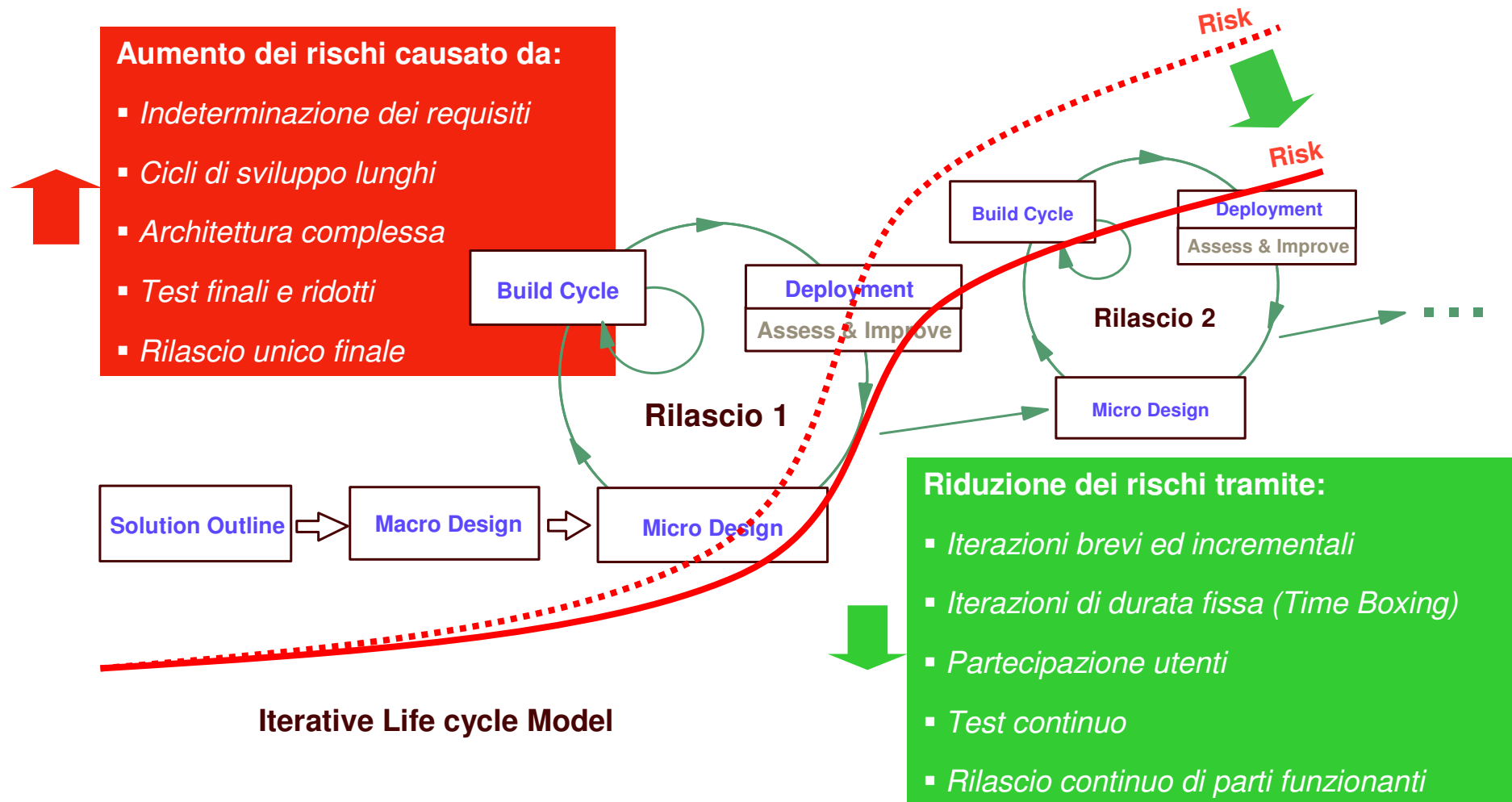
Water-fall Life-cycle

# Ciclo di vita iterativo

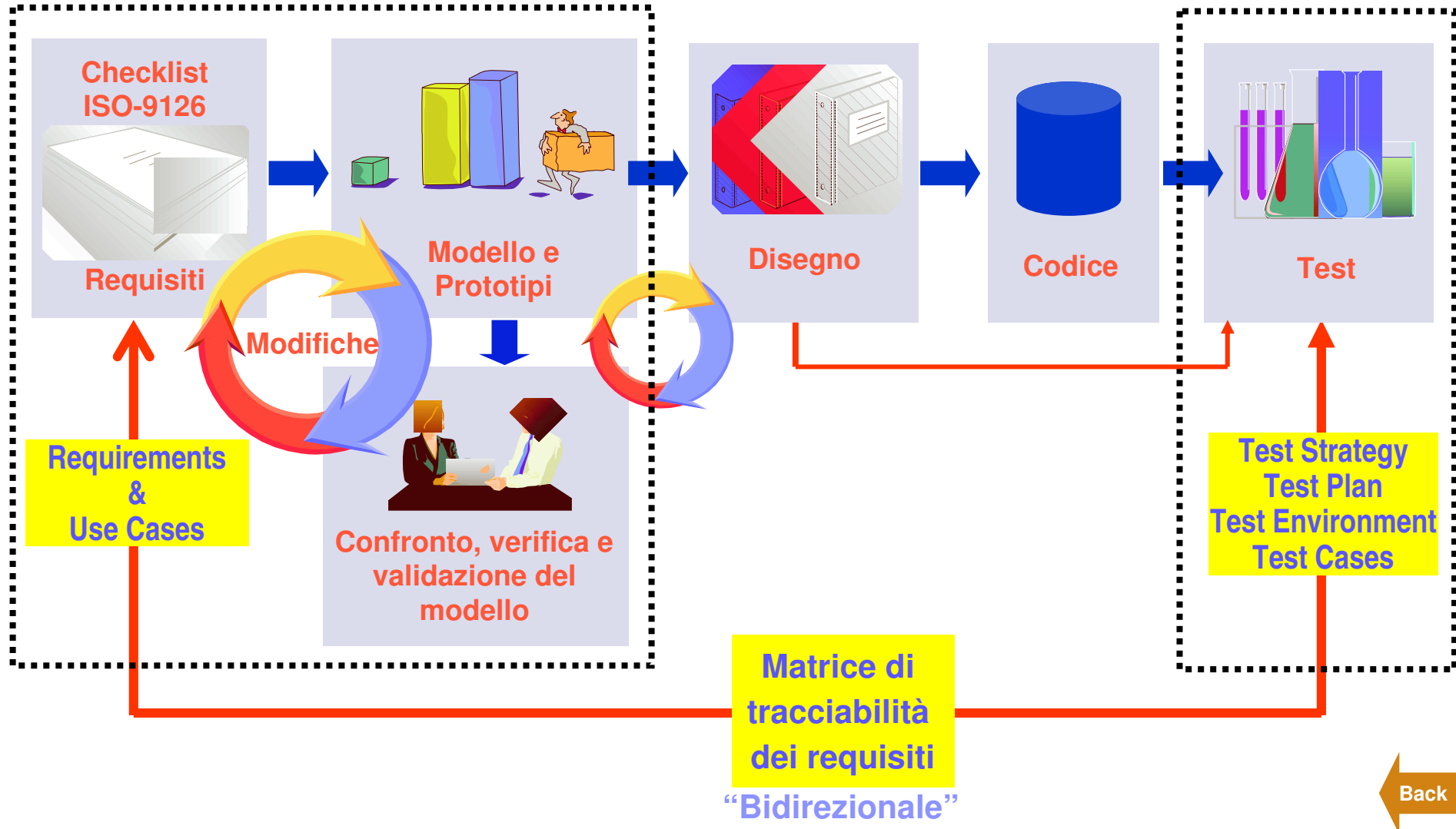


Iterative life-cycle

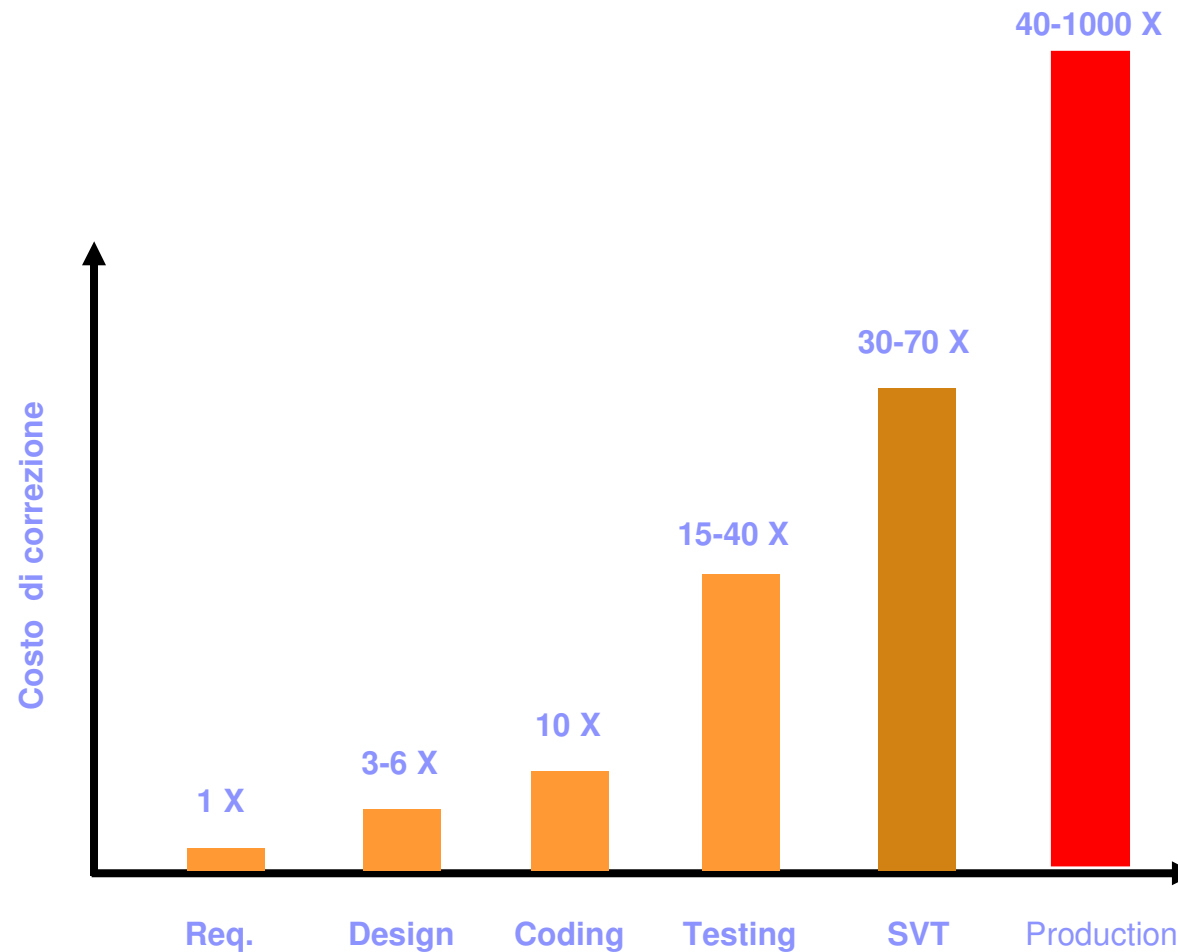
## Pro e contro del ciclo di vita “iterativo”



## In pratica ...

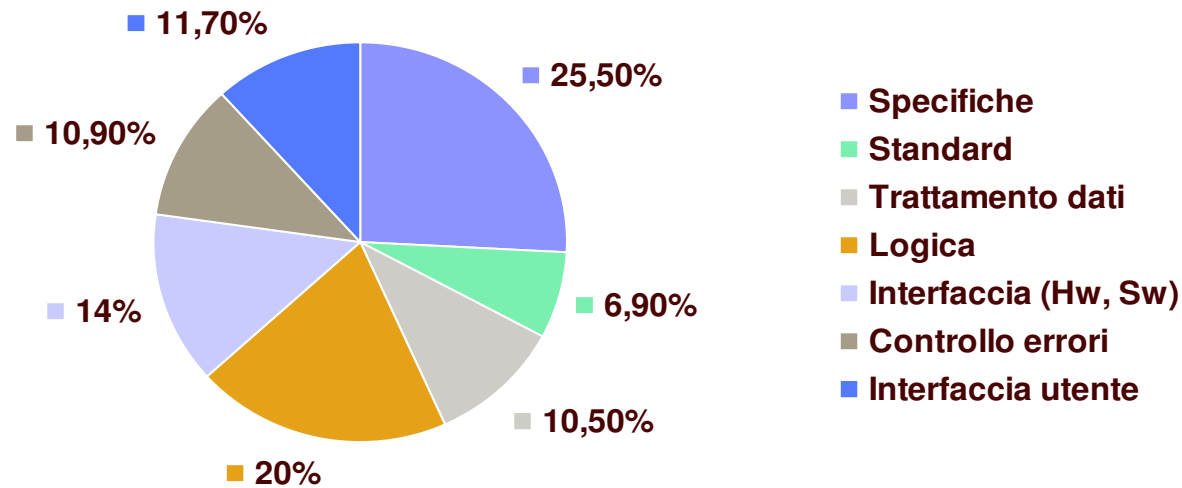


## Costo di rimozione degli errori nel software



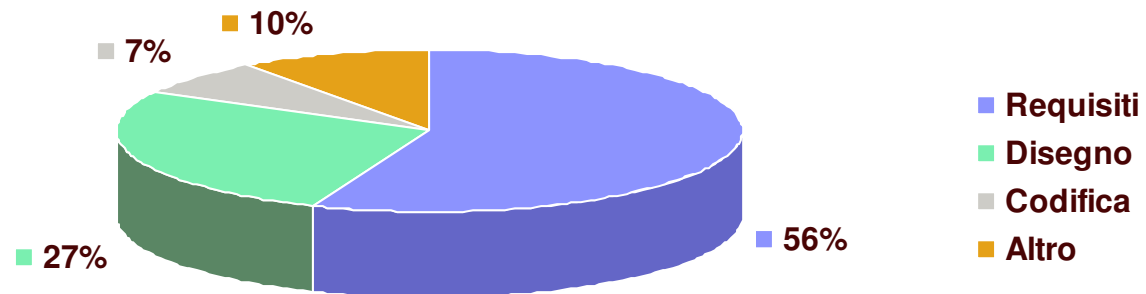
(Fonte: R.S. Pressman, 2000)

## Difettosità rilevata nelle diverse fasi di sviluppo del software



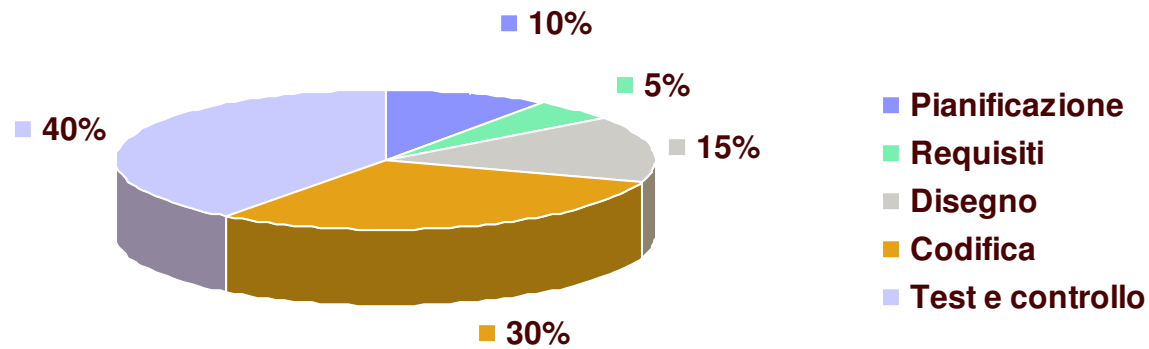
Cause e origine dei difetti di quattro progetti software, Grady, 1994

## Inserimento di difetti nel software per fase



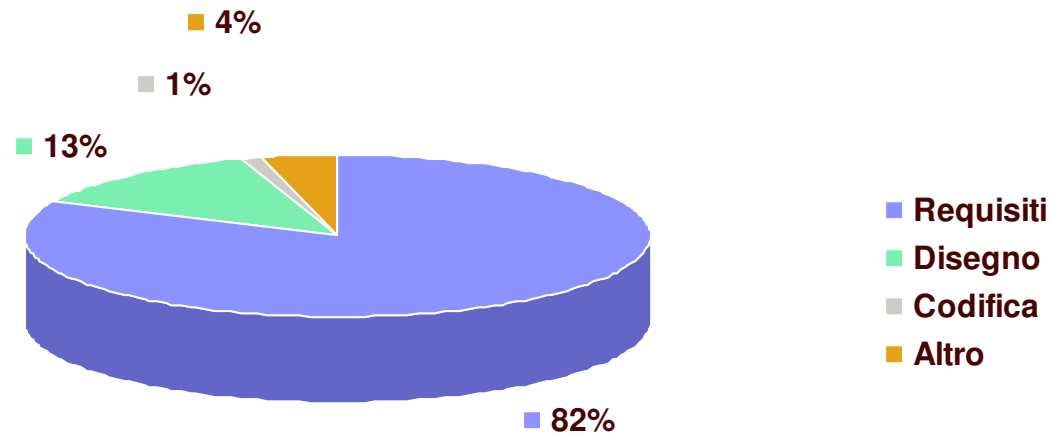
IBM, 1994

## Costo delle fasi dello sviluppo software



IBM, 1994

## Costo di rimozione degli errori per tipologia (fase di inserimento)



IBM, 1994

## *Verificare e valutazione della qualità del software*

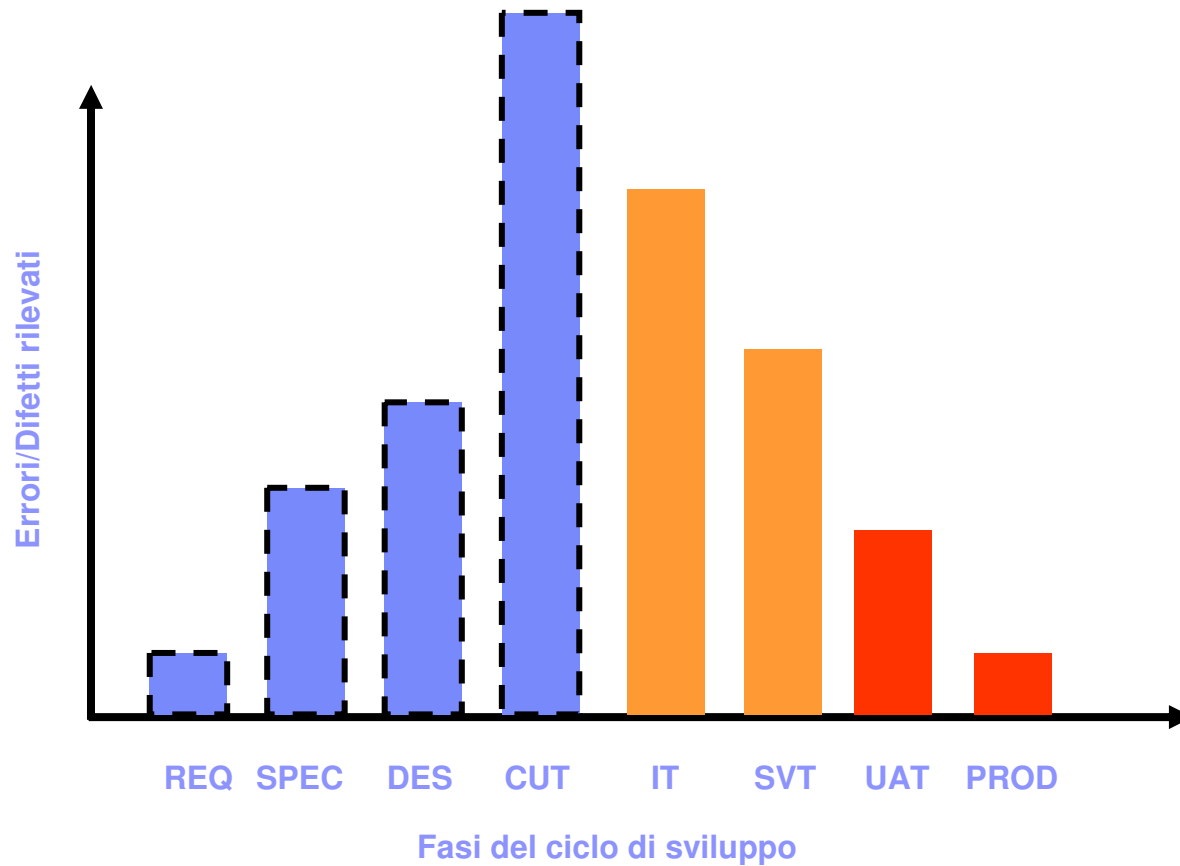
- *Profilo della difettosità del software*
- *Propagazione degli errori*
- *Revisioni tecniche*
- *Test*

*“Il 20% del codice contiene l’ 80% dei difetti.  
Trovateli e correggeteli.”*

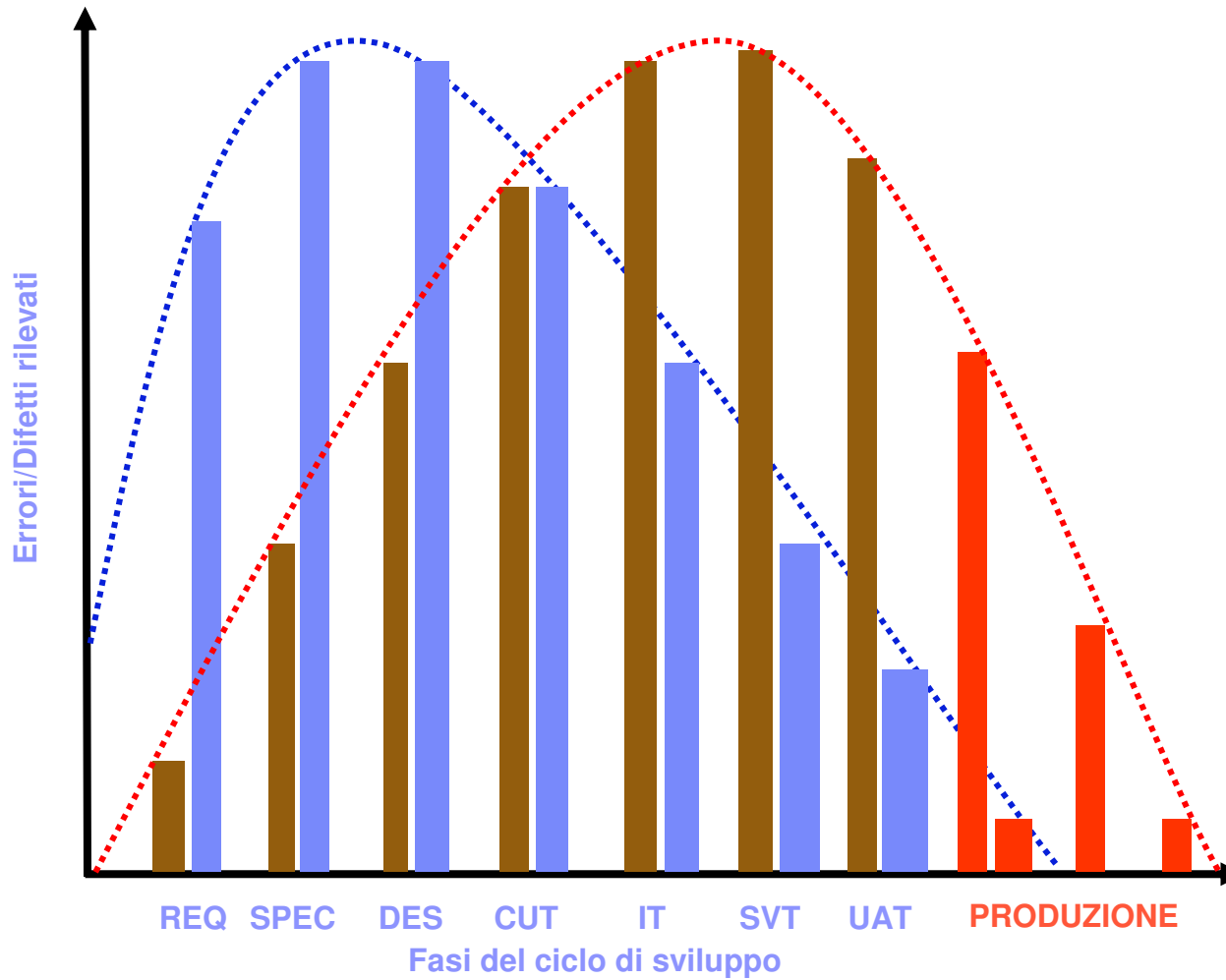
*Lowell Arthur*



## Profilo di rimozione degli errori nel software



## Curve di rimozione degli errori



## Cause più frequenti degli errori

DESCRIZIONE (TIPOLOGIA)	%
Specifiche incomplete o errate (IES)	22
Fraindimento delle comunicazioni del cliente (MCC)	17
Errore nella rappresentazione dei dati (EDR)	14
Collaudo incompleto o errato (IET)	10
Errore nella traduzione della progettazione nella codifica (PLT)	6
Inconsistenza dell'interfaccia tra i componenti (ICI)	6
Errore logico nella progettazione (EDL)	5
Deviazioni intenzionali dalle specifiche (IDS)	5
Documentazione imprecisa o incompleta (IID)	4
Violazione di standard di programmazione (VPS)	3
Interfaccia uomo-macchina ambigua o inconsistente (HCI)	3
Altro (MIS)	5
Totale	100

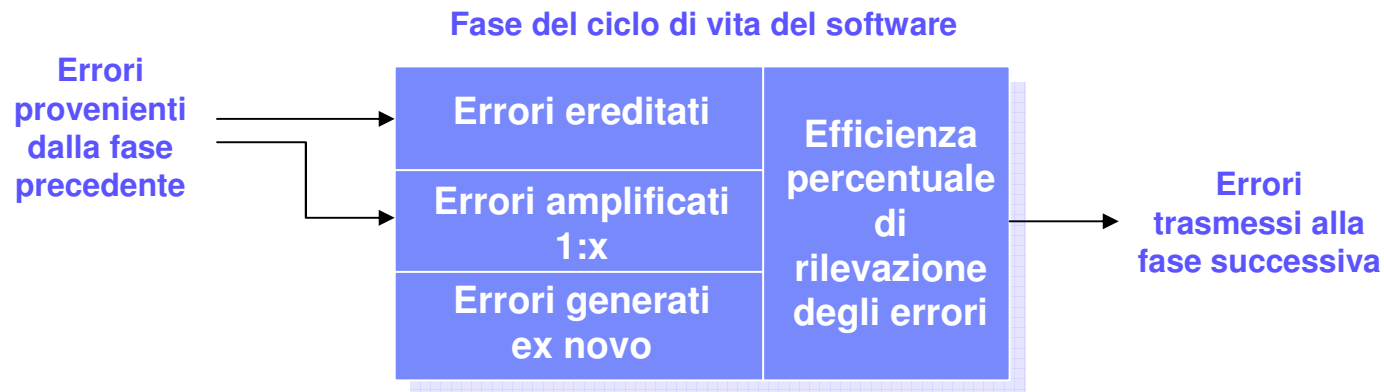


Source: Roger S. Pressman, 2000

# Propagazione degli errori

*“Alcune malattie, dicono i medici, all’inizio sono facili da curare ma difficili da riconoscere, ma con il passare del tempo, se non vengono riconosciute e trattate adeguatamente, divengono facili da riconoscere ma difficili da curare”*

*Nicolò Macchiavelli*



Source: IBM System Scientific Institute, 1981

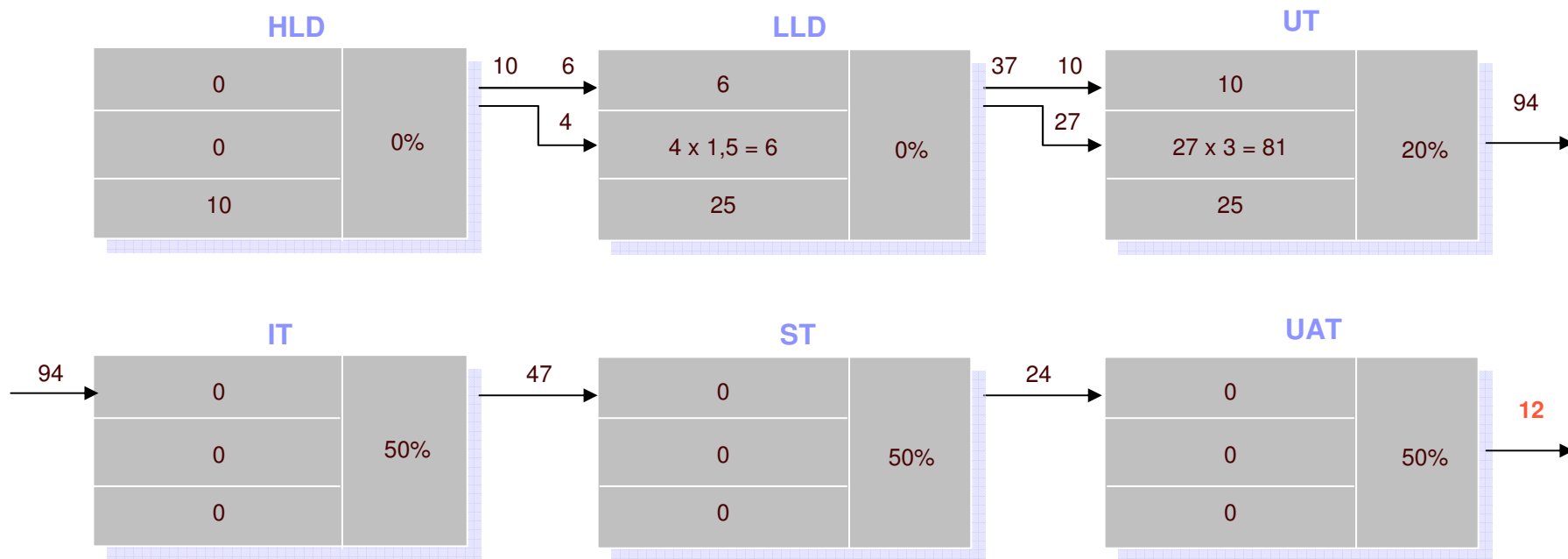
Costo della rimozione degli errori “senza” revisioni tecniche = 2177 unità di lavoro (errori residui = 12)

Costo della rimozione degli errori “con” revisioni tecniche = 780 unità di lavoro (errori residui = 3)



## Propagazione degli errori (senza revisioni)

Fase	Analisi e Disegno	Prima del collaudo	Durante il collaudo	Dopo il collaudo
Costo di rimozione	1	6,5	15	67

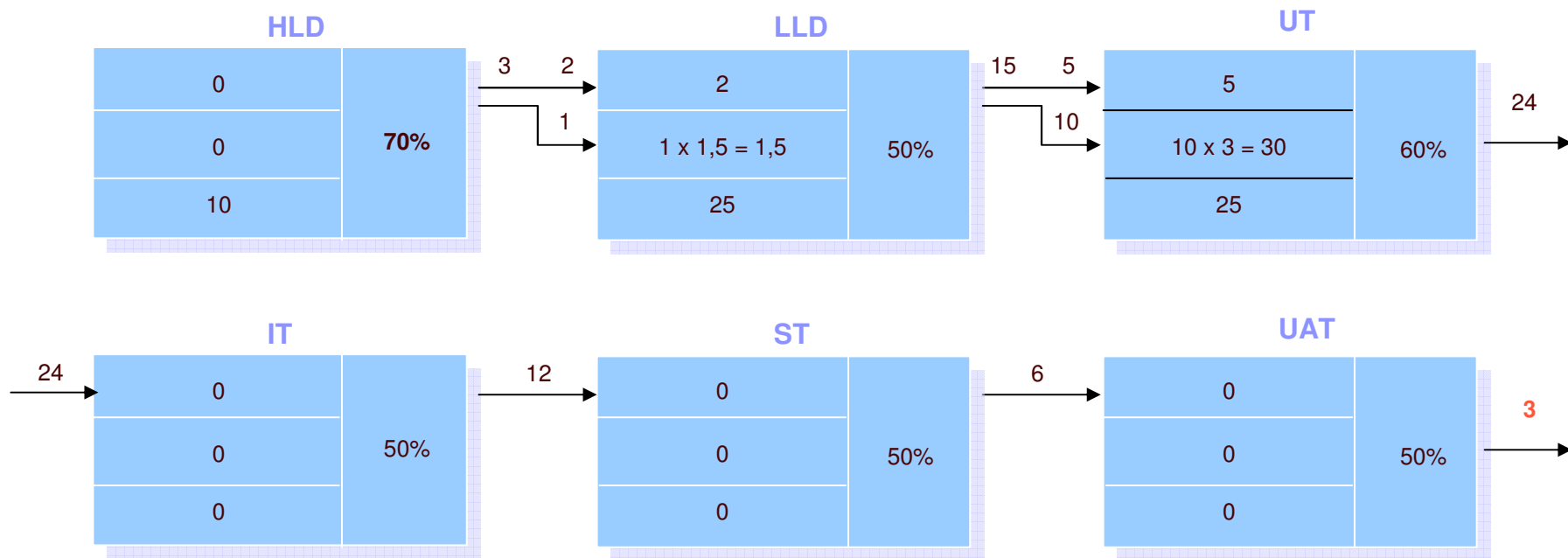


Source: IBM System Scientific Institute, 1981

**Costo totale  
rimozione errori =  
2177 unità di costo**

## Propagazione degli errori (con revisioni)

Fase	Analisi e Disegno	Prima del collaudo	Durante il collaudo	Dopo il collaudo
Costo di rimozione	1	6,5	15	67



Source: IBM System Scientific Institute, 1981

**Costo totale  
rimozione errori = 780  
unità di costo**

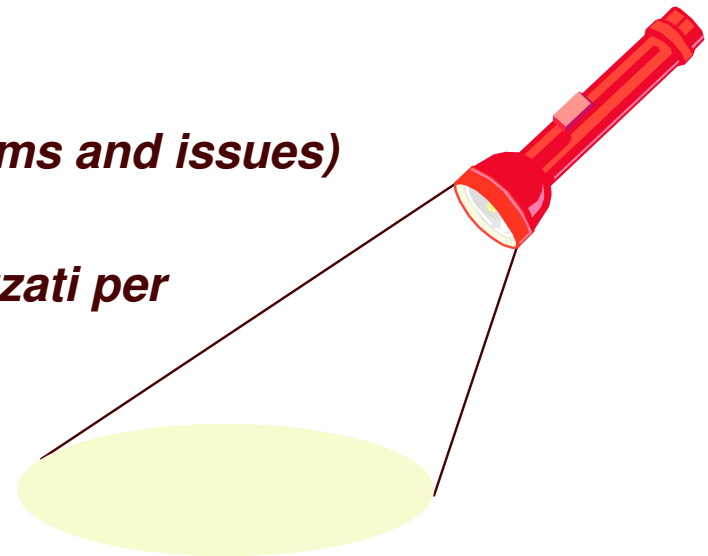
## Approccio alle revisioni tecniche del software

- Revisionare la documentazione più “critica” (piani, disegno, parti critiche di codice, casi di test, manuali etc.)
- Revisionare solo documenti “completati” o “parti rilevanti” di documenti completati
- Usare liste di controllo (checklist) costruite sull’esperienza
- Non spendere più di 2 ore per ciascuna revisione e 2 ore di preparazione (tot. 4 ore per ciascuna revisione)
- Registrare i risultati ed aggiornare le liste di controllo (checklist)
- Escludere i manager dalle revisioni tecniche
- Usare un approccio “Peer activity” (colleghi che si aiutano a vicenda)
- Fare attenzione nella scelta dei documenti (scegliere solo quelli importanti e critici per il progetto allo scopo di dedicare lo sforzo alle cose veramente utili e che fanno la differenza in termini di qualità)



## Le revisioni tecniche sono efficaci quando ...

- sono **pianificate** (*esse costano e devono quindi essere previste nel piano*)
- sono **preparate** (“No preparation no results”)
- sono **proattive** (*No judge, help*)
- **i risultati sono controllati** (*Solve problems and issues*)
- **la registrazione dei risultati sono analizzati per migliorare** (*Statistics, checklists*)



## Test del software

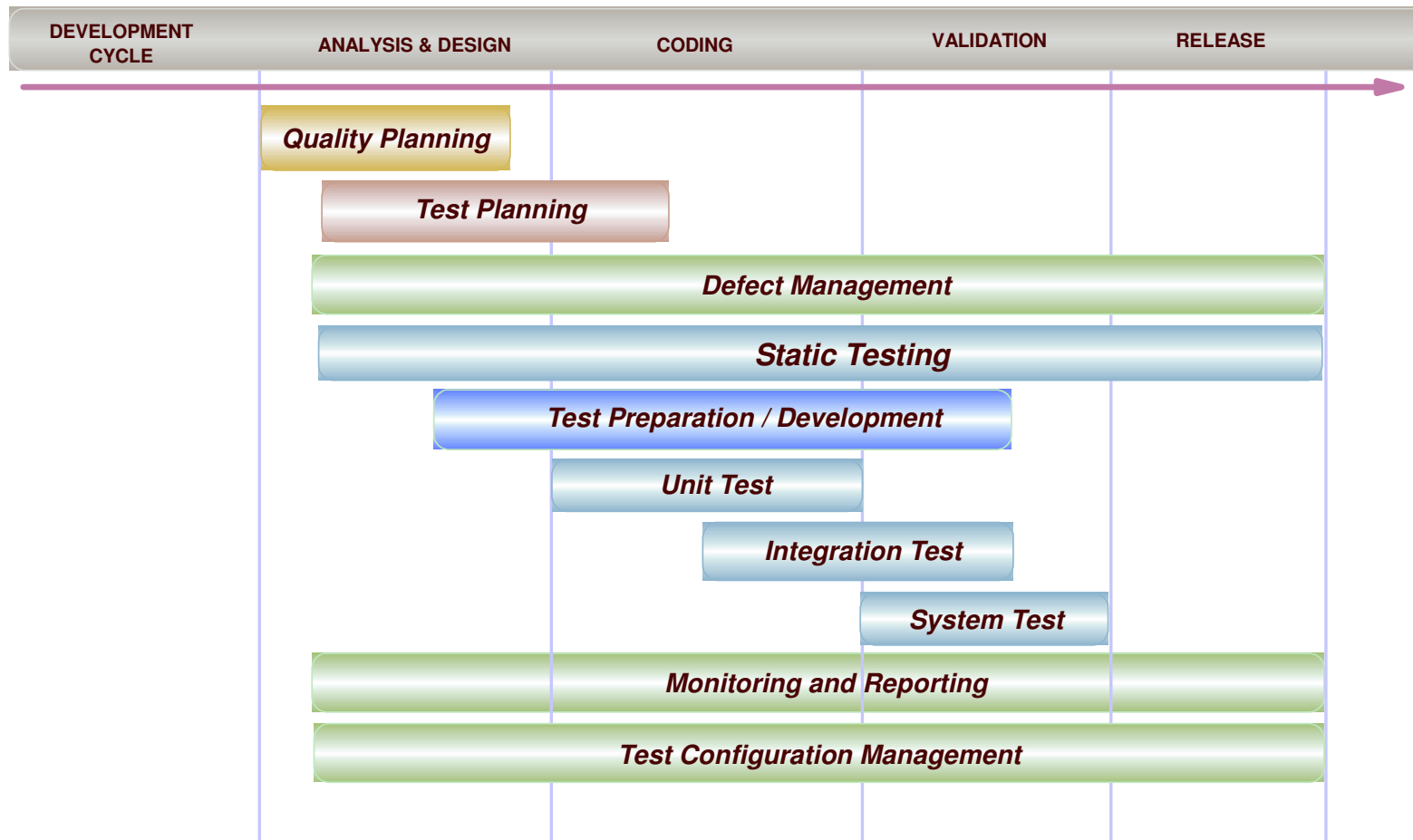
*“Lo sviluppo del software è un’attività produttiva ad alto contenuto umano dove le occasioni di commettere errori abbondano...”*

*... A causa dell’incapacità umana di comunicare in modo perfetto, lo sviluppo del software è affiancato da un’attività di garanzia di qualità.“*

Deutsch, 1979



# Modello di test continuo



Source: IBM Test Model

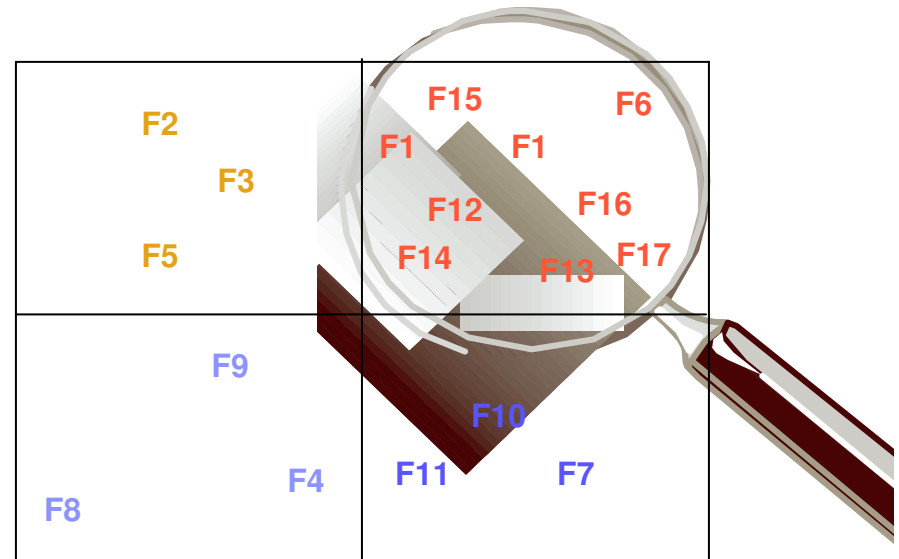
# Strategia di test

Quali e quanti test eseguire, quale livello di copertura occorre raggiungere, a quale livello di profondità occorre arrivare?

I test devono:

- *essere costruiti partendo dai requisiti e dai casi d'uso*
- *indirizzare sia le funzionalità previste sia le caratteristiche di qualità (performance, usabilità, robustezza, operatività, integrazione, portabilità etc.)*
- *verificare le “condizioni d'errore” previste e gestite*
- *verificare le “condizioni limite” e le condizioni “al contorno”*
- *avere una basedati progettata opportunamente (oracolo)*

Criticità della funzione per il business

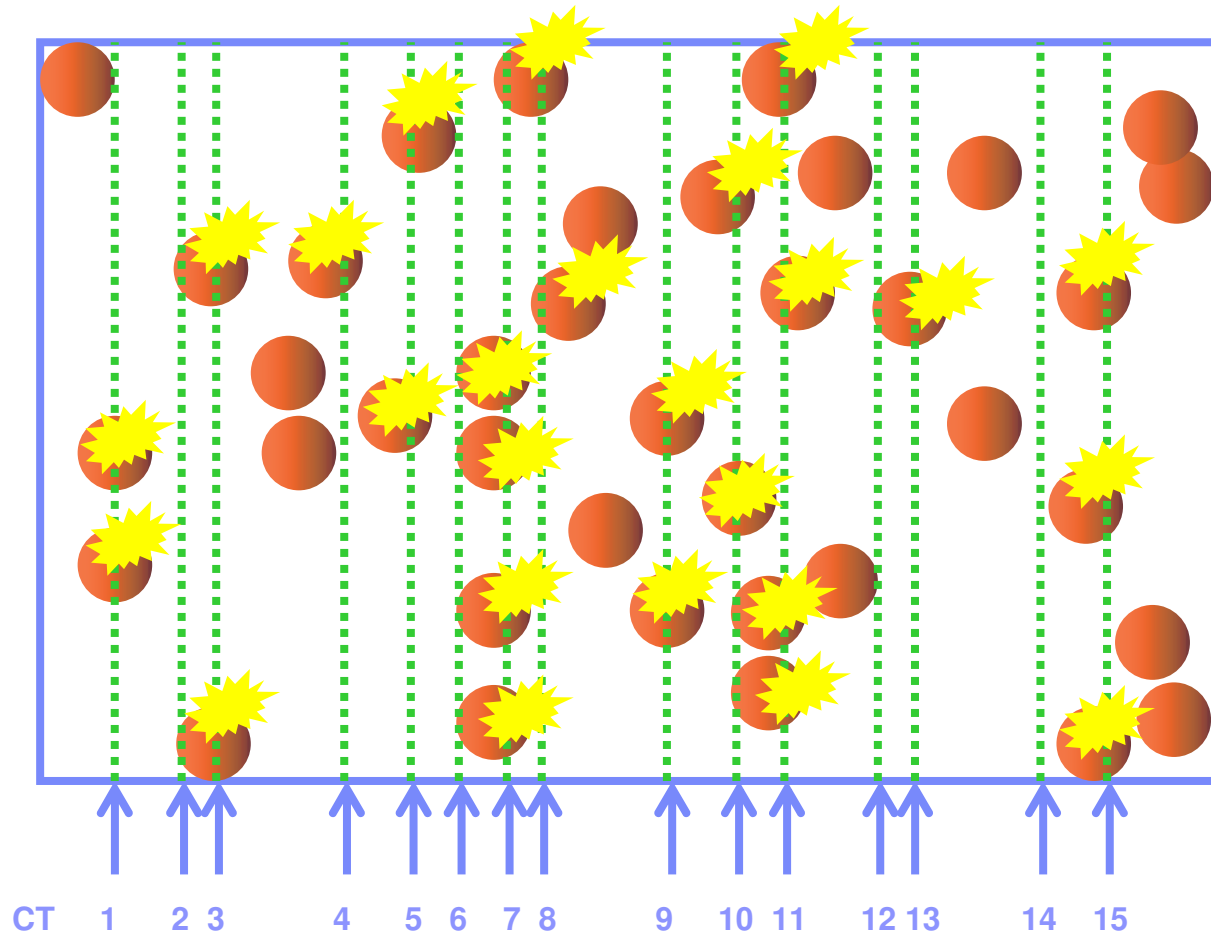


Frequenza di utilizzo della funzione

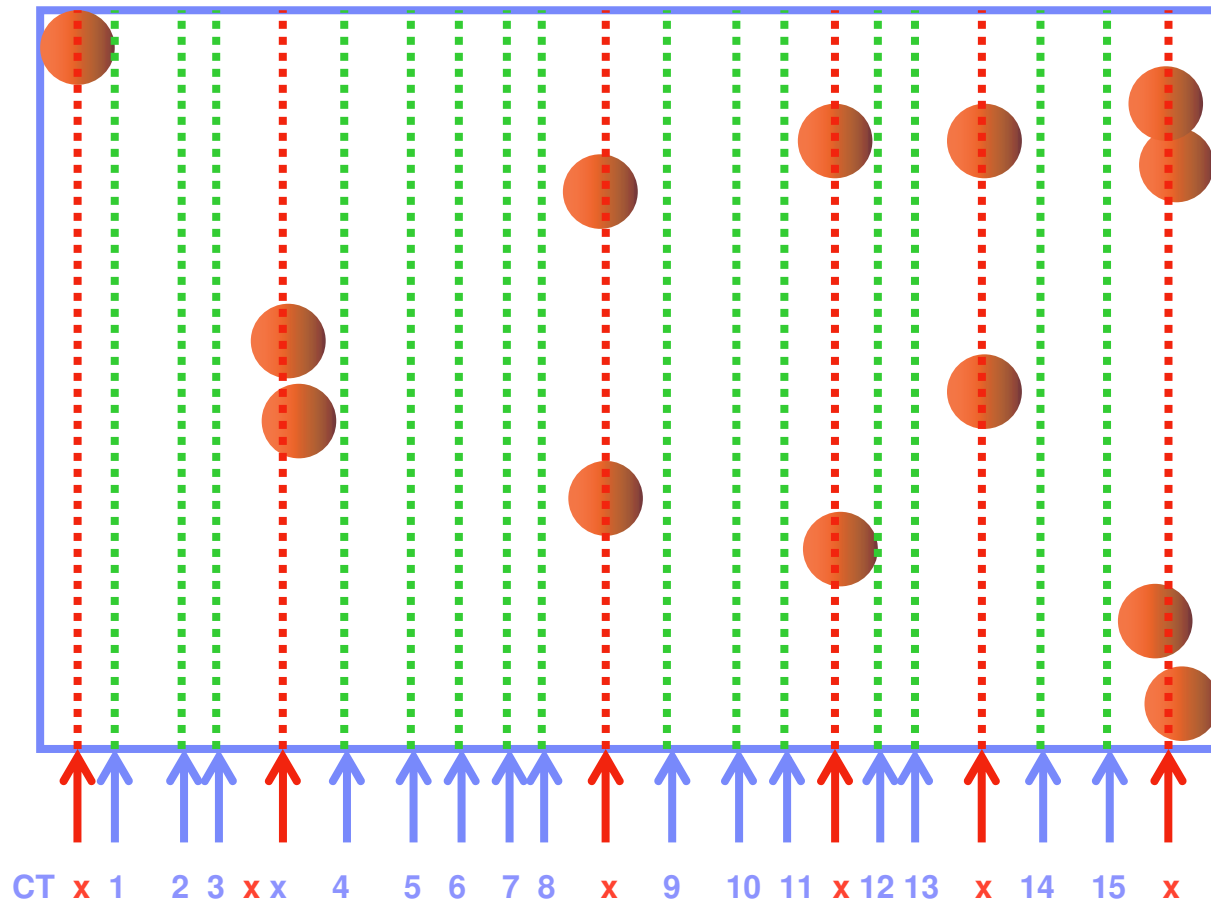
Automazione dei test

Riduzione dei rischi

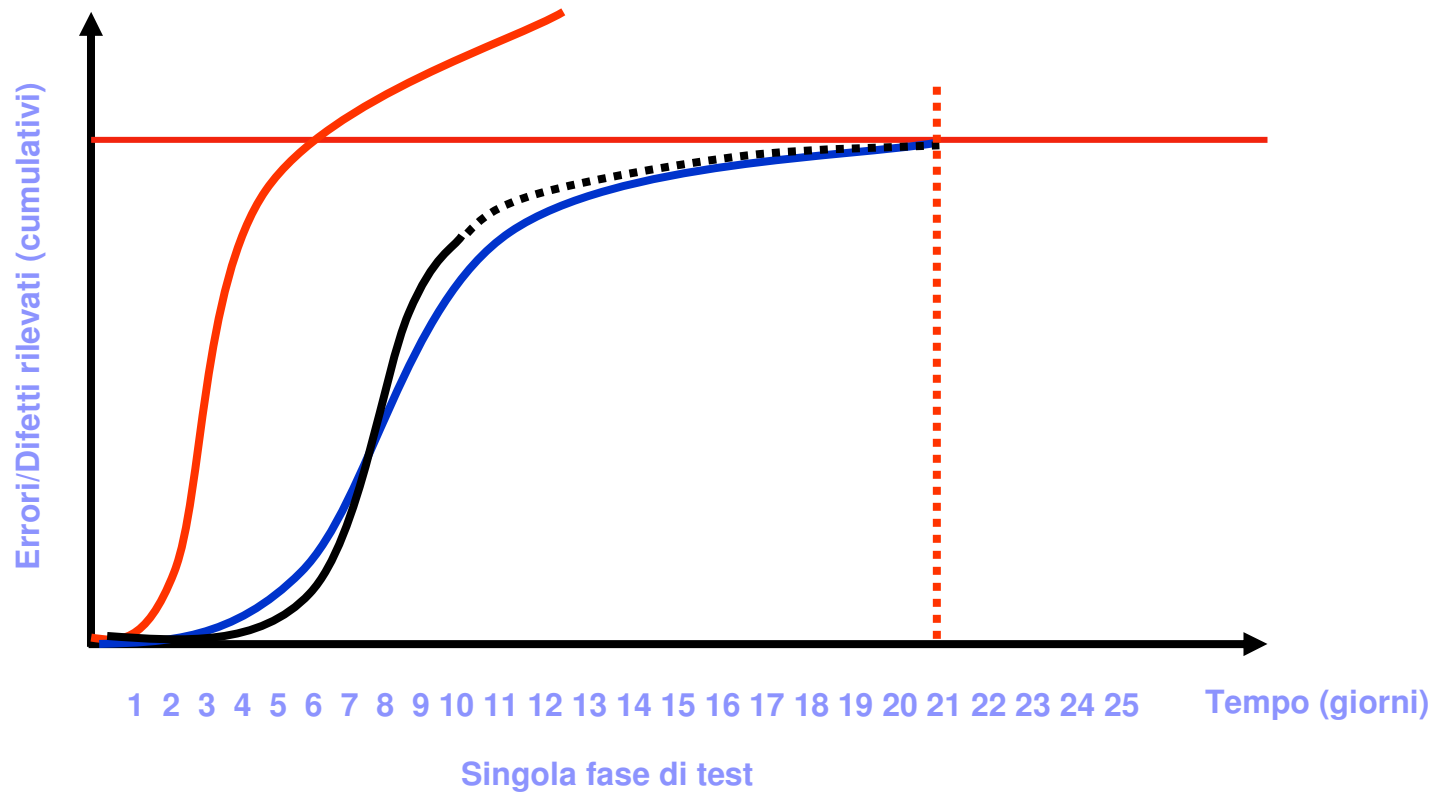
*“Quando può terminare una fase di test?”*



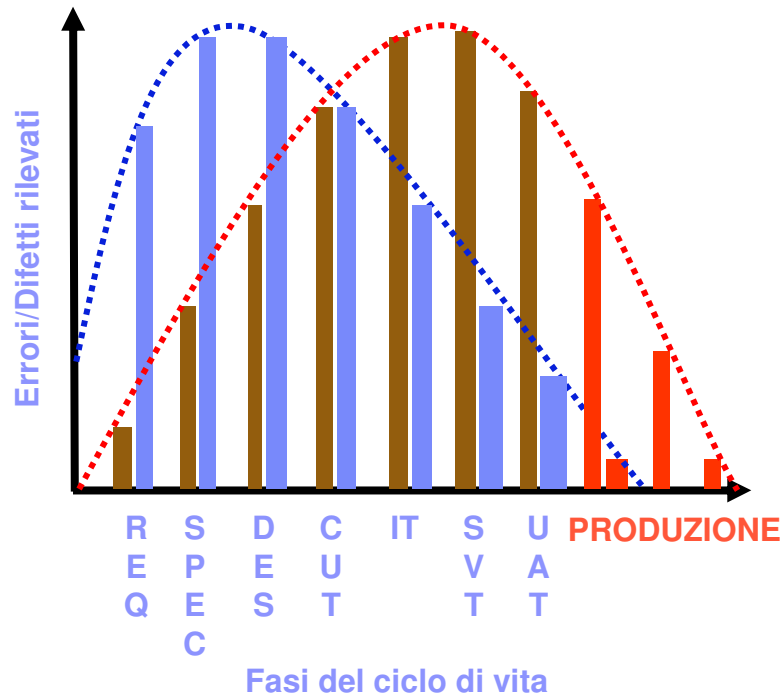
*“Quanti errori rimangono dopo la fase di test?”*



# Curva di rimozione degli errori durante la fase di test

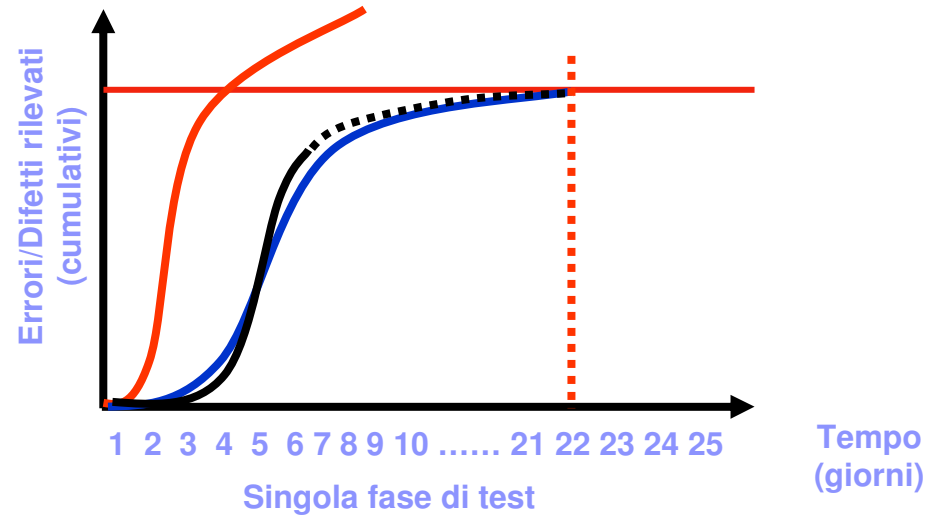


## Concludendo ...



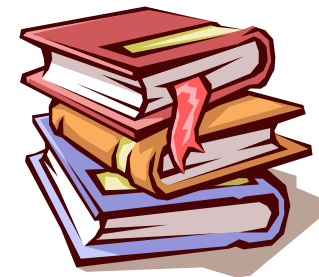
*“Occorre assolutamente conoscere il profilo dell’organizzazione che sviluppa il software in oggetto e ...*

*... occorre verificare la curva di rimozione degli errori nelle fasi di test e collaudo!”*



## Letture consigliate

- Roger S. Pressman – *Software Engineering, A Practitioner's Approach* – Mc Graw Hill (2000)
- Frederick P. Brooks, Jr . - *The Mythical Man-Month* - Addison-Wesley (1975, 1995 Anniversary Edition)
- Stefano Nocentini – *Il sistema di qualità del software* – ETASLIBRI (1993)
- Capability Maturity Model Integration, Version 1.1 – Staged Representation - SEI (2002)
- Glenford J. Myers - *The art of Software Testing* - Jhon Wiley & Sons, New York (1979)
- Steve McConnell – *Professional Software Development* - Addison-Wesley (2003)
- IEEE – *Guide to the Software Engineering Body of Knowledge* – Trial version – May 2001



## Siti web da visitare

*Project Management Center of Excellence (PM/COE): [http://w3-3.ibm.com/transform/project/home\\_0301.html](http://w3-3.ibm.com/transform/project/home_0301.html)*

*Easy-of-use: [http://www-306.ibm.com/ibm/easy/eou\\_ext.nsf/publish/558](http://www-306.ibm.com/ibm/easy/eou_ext.nsf/publish/558)*

*User-Centered Design (UCD): [http://eou5.austin.ibm.com/easy/eou\\_int.nsf/Publish/570](http://eou5.austin.ibm.com/easy/eou_int.nsf/Publish/570)*

*Software Engineering Institute (SEI): <http://www.sei.cmu.edu/sei-home.html>*

*Software Engineering Body of Knowledge (SWEBOK): <http://www.swebok.org/>*

