Verifica e validazione

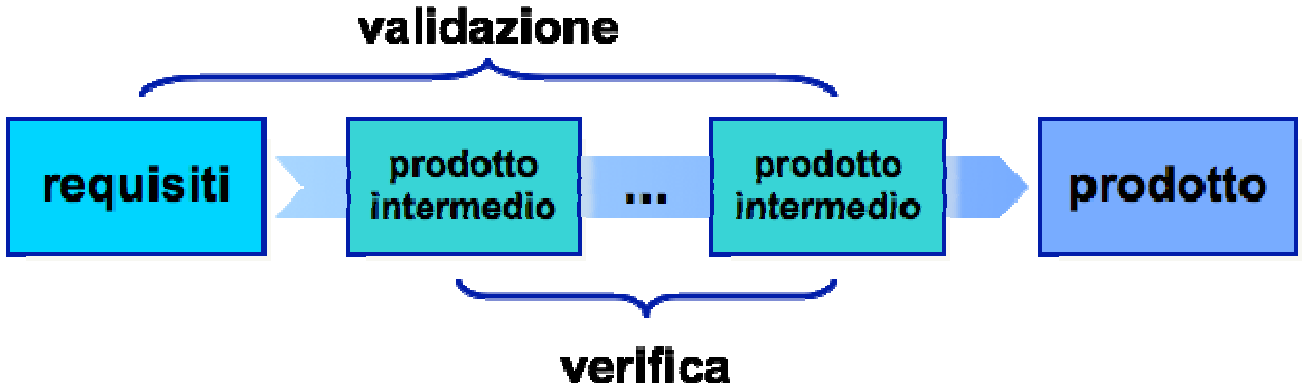
Vincenzo Gervasi, Laura Semini
Ingegneria del Software
Dipartimento di Informatica
Università di Pisa

Roadmap

- Concetti e terminologia
- Verifica, validazione, integrazione e collaudo
- Verifica statica
- Inspection e walkthrough

Verifica e validazione

- Attività necessarie :
  - per essere sicuri di non aver introdotto difetti
  - per essere sicuri di aver realizzato il prodotto giusto



Errori, difetti, malfunzionamenti

- Errore
  - causa (umana) — meccanico distratto
  - può non essere rilevante
- Difetto (bug) [errore in Binato et al.]
  - Introdotto da un errore —bullone avvitato male
  - Magari latente, da eliminare
- Malfunzionamento
  - prodotto da un difetto —perdita di una ruota
  - di solito è un danno

Controlli interni/esterni

- Interni alla azienda che ha sviluppato il sw
  - alfa
- Esterni
  - beta
  - collaudo

Verifiche statiche/dinamiche

- **Verifiche statiche**
  - **non prevedono l'esecuzione del codice**
  - usate specificamente per attività di verifica
  - eseguite sulle componenti
- **Verifiche dinamiche (prove/testing)**
  - **prevedono l'esecuzione del codice**
  - usate per attività di verifica o di validazione
  - eseguite sulle componenti o sul sistema

Test ideale

- Prova formale di correttezza
  - corrisponderebbe all'esecuzione del sistema con tutti i possibili input
  - "A convincing demonstration of correctness being impossible as long as the mechanism is regarded as a black box, our only hope lies in not regarding the mechanism as a black box", [Edsger W. Dijkstra, 1970]
  - è chiaramente impossibile
    - "Program testing can be used to show the presence of bugs, but never to show their absence!", idem

Risultati negativi

- *Teorema di Howden:*
  - Non esiste un algoritmo che, dato P qualsiasi, generi un test ideale finito (test definito da un criterio affidabile e valido)

- *Tesi di Dijkstra*
  - Il test di un programma può rilevare la presenza di malfunzionamenti, ma non dimostrarne l'assenza

Due soluzioni non ideali ma possibili

- Due soluzioni non ideali ma possibili:
  - verifiche statiche su modelli o con astrazioni dei dati in ingresso
  - verifiche dinamiche con “buona” selezione dei dati in ingresso

La verifica statica

- Verifica senza esecuzione del codice
- Metodi manuali
  - basati sulla lettura del codice (desk-check)
  - più comunemente usati
  - più o meno formalmente documentati
- Metodi formali or Statica supportata da strumenti
  - model checking
  - esecuzione simbolica

Perché la verifica statica

- Praticità e intuitività (desk-check)
- Ideale per alcune caratteristiche di qualità
- Convenienza economica
  - costi dipendenti dalle dimensioni del codice
  - bassi costi di infrastruttura (desk-check)
  - buona prevedibilità dei risultati

Metodi di lettura del codice

- Inspection e Walkthrough
- Sono metodi pratici
  - basati sulla lettura del codice
  - dipendenti dall'esperienza dei verificatori
  - per organizzare le attività di verifica
  - per documentare l'attività e i suoi risultati
- Sono metodi complementari tra loro

Inspection

- Obiettivi
  - rivelare la presenza di difetti
  - eseguire una lettura mirata del codice
- Strategia
  - focalizzare la ricerca su aspetti ben definiti (error guessing)
    - Ex: off-by-one error (aka Obi-Wan error)
- Agenti
  - verificatori diversi dai programmatori

Attività dell'inspection

- Fase 1: pianificazione
- Fase 2: definizione della lista di controllo
- Fase 3: lettura del codice
- Fase 4: correzione dei difetti

Walkthrough

- Obiettivo
  - rivelare la presenza di difetti
  - eseguire una lettura critica del codice
- Strategia
  - percorrere il codice simulandone l'esecuzione
- Agenti
  - gruppi misti ispettori e sviluppatori

Attività di walkthrough

- Fase 1: pianificazione
- Fase 2: lettura del codice
- Fase 3: correzione dei difetti

Inspection vs walkthrough

- Affinità
  - controlli statici basati su desk-test
  - programmatori e verificatori contrapposti
  - documentazione formale
- Differenze
  - inspection basato su errori presupposti
  - walkthrough basato sull'esperienza
  - walkthrough più collaborativo
  - inspection più rapido

Metodi formali per la verifica statica

- Model checking
- Esecuzone simbolica

Metodi formali per la verifica di sistemi

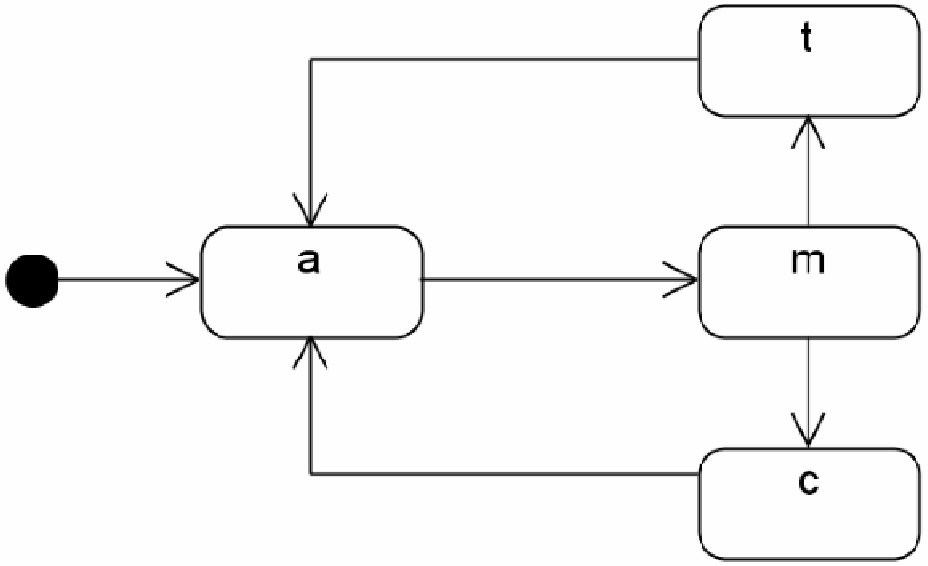
- Tecniche per descrivere a analizzare sistemi
  - basati su un formalismo matematico (logica, automi, teoria dei grafi)
- Una tecnica di verifica formale include:
  1. Una metodologia per modellare sistemi:
     - rappresentare un sistema in termini di oggetti matematici, astraendo e semplificandone la descrizione
  2. Un linguaggio di specifica, per descrivere le proprietà del sistema modellato.
     - Ex: logica temporale
  3. Un metodo di verifica:
     - tecniche di analisi formale per verificare se un sistema soddisfa la sua specifica.

Model checking

- "Given a model of a system, exhaustively and automatically check whether this model meets a given specification" [wiki]
- ~1980 per circuiti hw
  - circuito modellato con macchine a stati finiti
- In seguito applicato anche (soprattutto) al sw
- Specifica comportamento (modello)
  - normalmente con sistemi di transizioni (nodi/archi)
- Proprietà specificate con logica temporale
  - A. Pnueli, E. M. Clarke, E. A. Emerson and J. Sifakis

Model checking: un modello

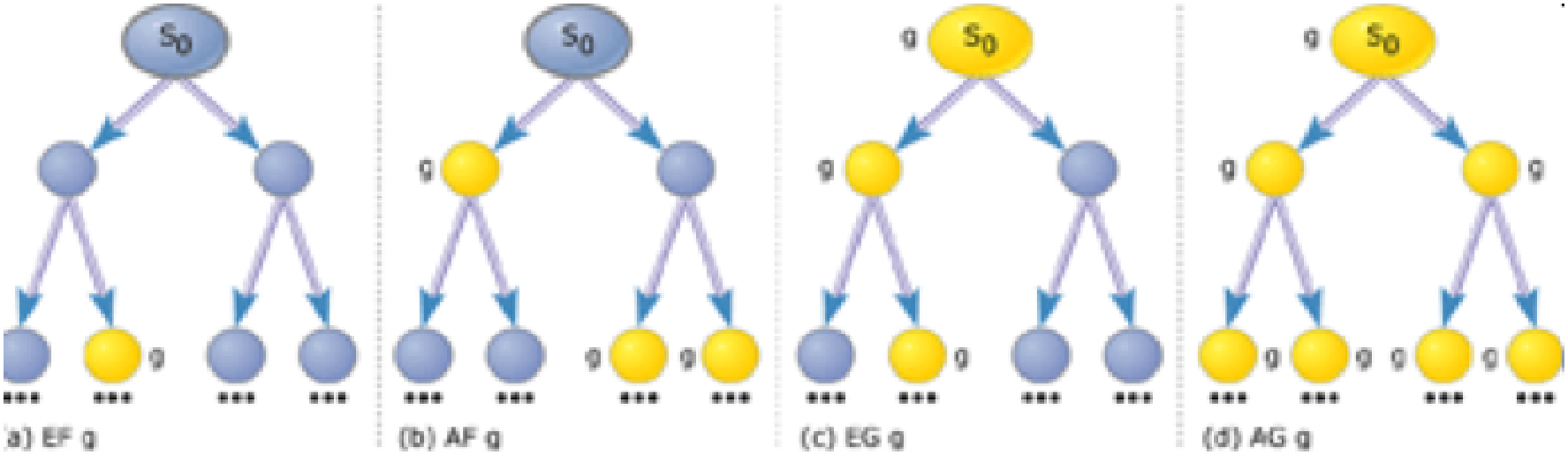
- Inizialmente in stato di attesa (a)
- Dopo l'inserimento di una moneta si passa a uno stato "moneta inserita" (m)
- Infine è consegnato un caffè (c) o un tè (t)



Alcune proprietà

- È sempre vero che se è stata inserita una moneta, allora in tutti i cammini del modello si incontra prima o poi uno stato nel quale viene consegnato un tè o uno stato nel quale viene consegnato un caffè
- Non vengono mai consegnati allo stesso tempo sia il tè che il caffè
- È sempre vero che se è stata inserita una moneta, allora in tutti i cammini del modello si incontra prima o poi uno stato nel quale viene consegnato un tè

Alcuni operatori temporali



Formalizzazione delle proprietà

- È sempre vero che se è stata inserita una moneta, allora in tutti i cammini del modello si incontra prima o poi uno stato nel quale viene consegnato un tè o uno stato nel quale viene consegnato un caffè
  - AG [m → AF (c OR t)]
- Non vengono mai consegnati allo stesso tempo sia il tè che il caffè
  - AG ~ (c AND t)
- È sempre vero che se è stata inserita una moneta, allora in tutti i cammini del modello si incontra prima o poi uno stato nel quale viene consegnato un tè
  - AG [m → AF t]

1986: algoritmo Clarke Emerson Sistla

- Data una specifica M e una formula f, si verifica se M soddisfa f ($M \models f$), cioè, se M è un modello per f
- Se la risposta è negativa, di solito viene prodotto un contro-esempio, ovvero, un cammino nel modello che non verifica la (sotto)formula
- L'algoritmo attraversa lo spazio degli stati etichettandoli con sotto-formule, ed è quadratico rispetto al numero degli stati (lineare in varianti)

Problema

- State explosion
  - numero di stati esponenziale nel numero dei processi coinvolti
  - algoritmi di MC classici non vanno oltre il milione di stati
  - possibile soluzione: interleaving vs. true concurrency

Esecuzione simbolica

- L'esecuzione di un programma viene simulata
  - si rappresentano insiemi di possibili dati d'ingresso con simboli algebrici, piuttosto che con valori numerici reali

Esec. simbolica: un esempio

```
public static int abs (int i) {

    int r;                          //L1

    if (i<0)
            r = -i;                 //L2
    else
            r = i;                  //L3
                                    //L4
            return r;
    }
```

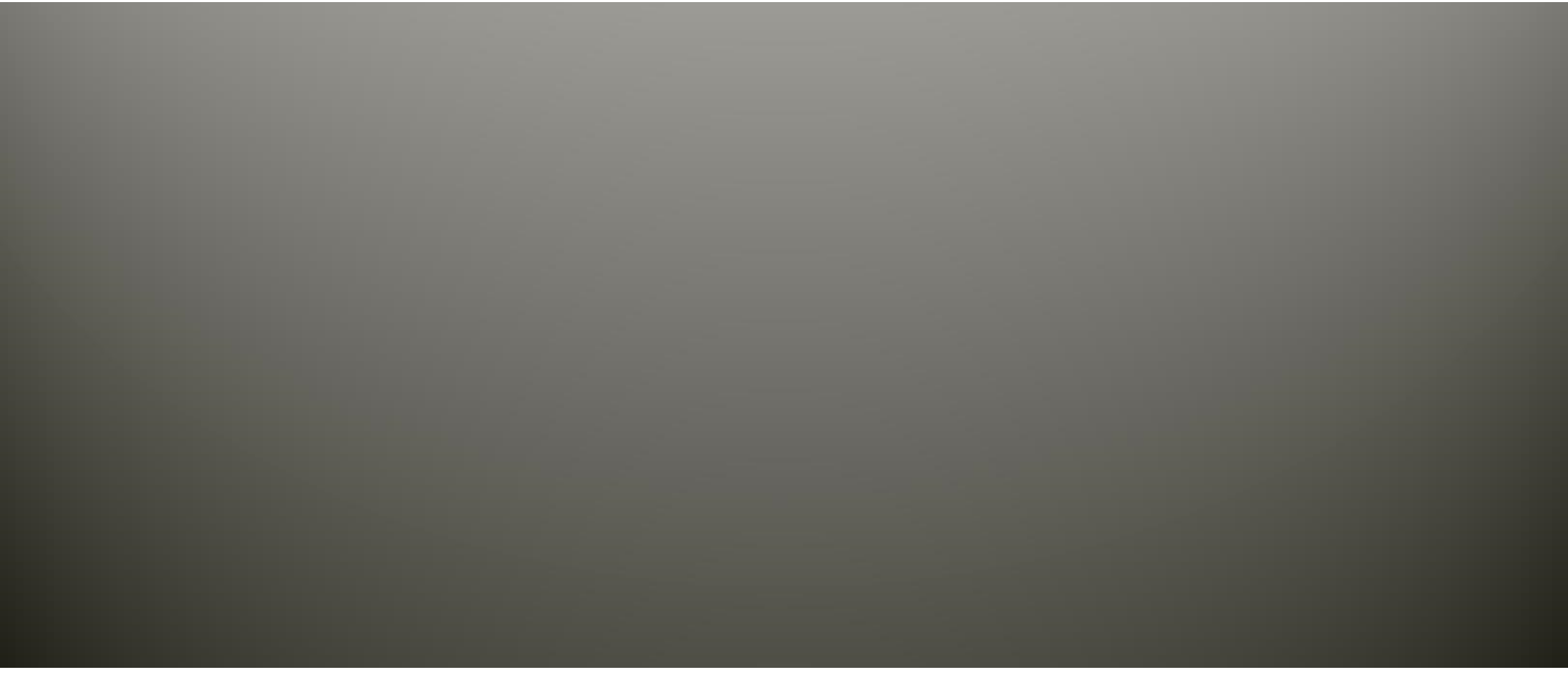
Esec. simbolica: un esempio ctn'd

- L'esecuzione simbolica simula l'esecuzione, associando a ogni stato un'espressione che ne descrive le proprietà, in funzione dei valori assunti dalle variabili

passi	stato	note
L1	$i = I$	è un intero
L2	$i = I \ \& \ (I < 0 \ \& \ r = -I)$	
L3	$i = I \ \& \ (0 \leq I \ \& \ r = I)$	
L4	$i = I \ \& \ ((I < 0 \ \& \ r = -I) \mid (0 \leq I \ \& \ r = I))$	I rami del condizionale si ricongiungono

- In generale, la valutazione dei predicati di questo genere è un problema non decidibile. In tanti casi però le espressioni simboliche possono essere semplificate, ed i risultati come quello appena visto possono essere ottenuti da un theorem prover

Verifica dinamica o testing



Verifica dinamica o testing

- Si compone di più fasi:
  - Progettazione (input, ma non solo)
  - Definizione ambiente di test
  - Esecuzione del codice
  - Analisi dei risultati (output ottenuto con l'esecuzione vs output atteso)
  - Debugging

Proprietà e aspetti del testing

- Ripetibilità
- Verifica di componenti vs verifica di sistema
- Test di integrazione
- Vari tipi di test sul Sistema
- Test di accettazione (o collaudo)

Ripetibilità

- Ripetibilità della prova
  - Ambiente definito (hardware, condizioni, …)
  - casi di prova definiti (ingressi e comportamenti attesi)
  - procedure definite
- Strumenti
  - Driver componente fittizia per pilotare un modulo
  - Stub componente fittizia per simulare un modulo
- Registrazione e analisi dei dati di prova

Verifica di componenti e sistema

- Verifiche sulle componenti
  - approccio statico, dal desk-check all'inspection
  - approccio dinamico, con realizzazione di driver e stub
- Verifiche sul sistema
  - approccio dinamico, sul sistema completo
  - approccio dinamico, durante l'integrazione, con la realizzazione
    di driver e stub

Integrazione

- Metodologie diverse (già discusse)
  - big bang, bottom up, top down, sandwich
- Stub e driver
  - costruiti usando il grafo della relazione di uso dei moduli

test sul sistema, 1

- Facility test (test delle funzionalità)
  - è il più intuitivo dei controlli, quello cioè che mira a controllare che ogni funzionalità del prodotto stabilita nei requisiti sia stata realizzata correttamente
- Security test
  - cercando di accedere a dati o a funzionalità che dovrebbero essere riservate, si controlla l'efficacia dei meccanismi di sicurezza del sistema

test sul sistema, 2

- Usability test
  - si valuta la facilità d'uso del prodotto da parte dell'utente finale
    - valutazione fra le più soggettive
    - prodotto + documentazione + livello di competenza dell'utenza + caratteristiche operative dell'ambiente d'uso del prodotto
- Performance test
  - controllo mirato a valutare l'efficienza di un sistema soprattutto rispetto ai tempi di elaborazione e ai tempi di risposta
    - controllo critico per esempio per i sistemi in tempo reale, per i quali ai requisiti funzionali si aggiungono rigorosi vincoli temporali
    - l sistema viene testato a diversi livelli di carico, per valutarne le prestazioni

test sul sistema, 3

- Storage use test
  - ancora un controllo legato all'efficienza di un sistema, ma mirato alla richiesta di risorse – la memoria in particolare – durante il funzionamento, e ha implicazioni sull'ambiente operativo richiesto per poter installare il sistema
- Volume test (o load test, test di carico)
  - il sistema è sottoposto al carico di lavoro massimo previsto dai requisiti e le sue funzionalità sono controllate in queste condizioni
    - lo scopo è sia individuare malfunzionamenti che non si presentano in condizioni normali, quali difetti nella gestione della memoria, buffer overflows, etc., sia garantire un'efficienza base anche in condizioni di massimo carico
    - le tecniche e gli strumenti come per il performance test: i due tipi di test hanno scopi molto differenti, da un lato valutare le prestazioni a vari livelli di carico, non limite, dall'altro valutare il comportamento del sistema sui valori limite

test sul sistema, 4

- Stress test
  - il sistema è sottoposto a carichi di lavoro superiori a quelli previsti dai requisiti o è portato in condizioni operative eccezionali – in genere sottraendogli risorse di memoria e di calcolo
    - esplicito superamento dei limiti operativi previsti dai requisiti
    - lo scopo è quello di controllare la capacità di "recovery" (recupero) del sistema dopo un fallimento
- Configuration test
  - obiettivo: la prova del sistema in tutte le configurazioni previste
    - piattaforme di installazione diverse per sistema operativo o dispositivi hardware installati
    - insiemi di requisiti funzionali leggermente diversi

test sul sistema, 5

- Compatibility test
  - obiettivo: valutare la compatibilità del sistema con altri prodotti software
    - versioni precedenti dello stesso prodotto
    - sistemi diversi, ma funzionalmente equivalenti che il prodotto deve rimpiazzare
    - altri sistemi software con i quali il prodotto deve interagire

Test di accettazione (se contrattualizzato: collaudo)

- Validazione del sistema
  - attività svolta dal fornitore
    - alfa test o pre-collaudo
  - attività svolta dal committente o da terze parti
    - Collaudo o beta test
  - su casi di prova definiti nel contratto
- Valore contrattuale
  - il collaudo è un'attività ufficiale
  - conclusione della commessa (a meno di servizi e garanzie)
  - al collaudo segue il rilascio del sistema