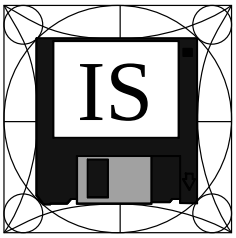
Progettazione delle prove



IS 2001-4
Corso di Ingegneria del Software

V. Ambriola, G.A. Cignoni,
C. Montangero, L. Semini

Con aggiornamenti di: T. Vardanega

Contenuti

- Caratteristiche delle prove del *software*
- Criteri funzionali
- Criteri strutturali
- Analisi dei risultati, l'oracolo
- Approfondimento: valutazione delle prove

Prova e collaudo

- Analisi dinamica
  - Attività che prevedono l'esecuzione del *software*
  - In un ambiente controllato e con ingressi ed uscite definiti
  - Come verifica (prova sui moduli)
  - Come validazione (collaudo sul sistema)
- Metodo intuitivo ma complesso
  - Progettazione, esecuzione, analisi, correzione dei difetti
  - Validazione dei risultati, terminazione delle prove

Caratteristiche delle prove

- Una prova non è sempre definitiva
  - Deve essere ripetibile (dunque ben specificata)
  - I risultati prodotti valgono solo per la prova effettuata
    - Non possono essere generalizzati
  - Rileva malfunzionamenti (indicando la presenza di guasti)
  - Non può provarne l'assenza!
- Le prove sono costose
  - Richiedono molte risorse (tempo, persone, infrastrutture)
  - Necessitano di un processo definito
  - Richiedono attività di ricerca del difetto, analisi e correzione

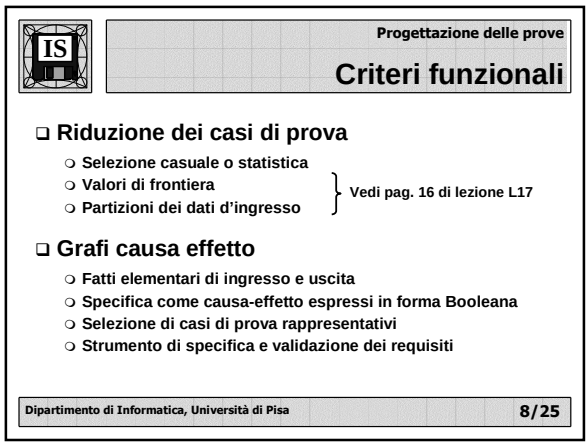
Gli elementi di una prova

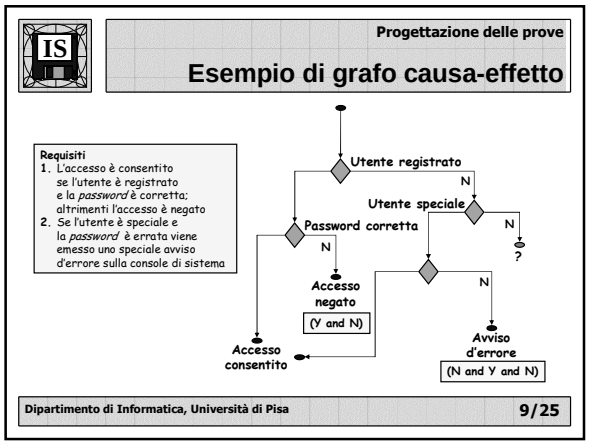
- Caso di prova (*test case*)
  - Una tripla <ingresso, uscita, ambiente>
    - L'ambiente include l'oggetto della prova
- Batteria di prove (*test suite*)
  - Un insieme (sequenza) di casi di prova
- Procedura di prova
  - Il procedimento (automatico e non) per eseguire, registrare analizzare e valutare i risultati di una batteria di prove

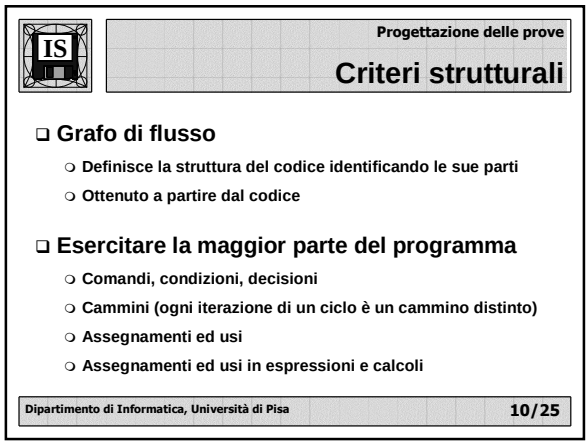
Conduzione di una prova

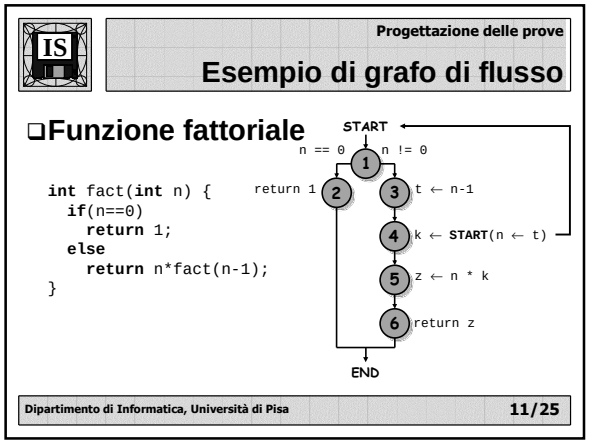
- Definizione dell'obiettivo della prova
- Progettazione della prova
- Realizzazione dell'ambiente di prova
- Esecuzione della prova
- Analisi dei risultati
- Valutazione della prova


IS
Progettazione delle prove
Progettazione
Criteri funzionali
A scatola chiusa (black box)
Basati sulla "conoscenza" dei requisiti di qualità
Funzionalità, affidabilità, efficienza
Criteri strutturali
A scatola aperta (white box)
Basati sulla "conoscenza" del codice
L'obiettivo primario è esercitare tutto il software del prodotto
Combinazione delle strategie → gray box
Dipartimento di Informatica, Università di Pisa
7/25

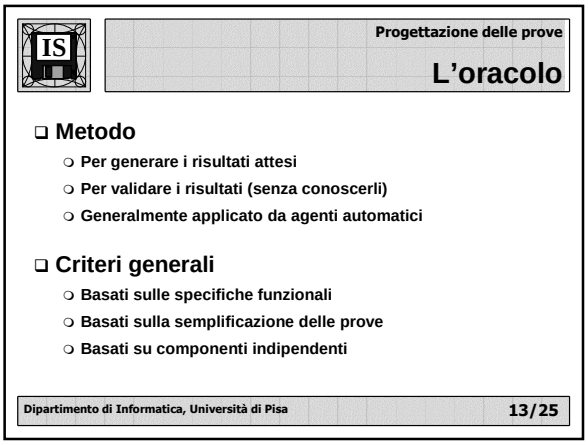

IS
Progettazione delle prove
Criteri funzionali
Riduzione dei casi di prova
Selezione casuale o statistica
Valori di frontiera
Partizioni dei dati d'ingresso
Vedi pag. 16 di lezione L17
Grafi causa effetto
Fatti elementari di ingresso e uscita
Specifica come causa-effetto espressi in forma Booleana
Selezione di casi di prova rappresentativi
Strumento di specifica e validazione dei requisiti
Dipartimento di Informatica, Università di Pisa
8/25


IS
Progettazione delle prove
Esempio di grafo causa-effetto
Requisiti
1. L'accesso è consentito se l'utente è registrato e la password è corretta; altrimenti l'accesso è negato
2. Se l'utente è speciale e la password è errata viene emesso uno speciale avviso d'errore sulla console di sistema
Utente registrato
N
Utente speciale
N
Password corretta
N
?
N
Accesso negato
(Y and N)
Accesso consentito
Avviso d'errore
(N and Y and N)
Dipartimento di Informatica, Università di Pisa
9/25

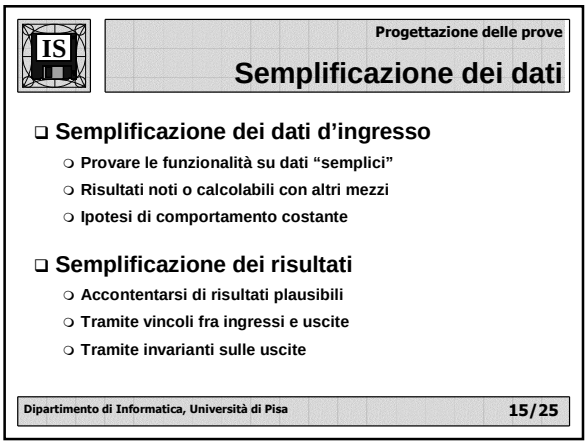

IS
Progettazione delle prove
Criteri strutturali
Grafo di flusso
Definisce la struttura del codice identificando le sue parti
Ottenuto a partire dal codice
Esercitare la maggior parte del programma
Comandi, condizioni, decisioni
Cammini (ogni iterazione di un ciclo è un cammino distinto)
Assegnamenti ed usi
Assegnamenti ed usi in espressioni e calcoli
Dipartimento di Informatica, Università di Pisa
10/25

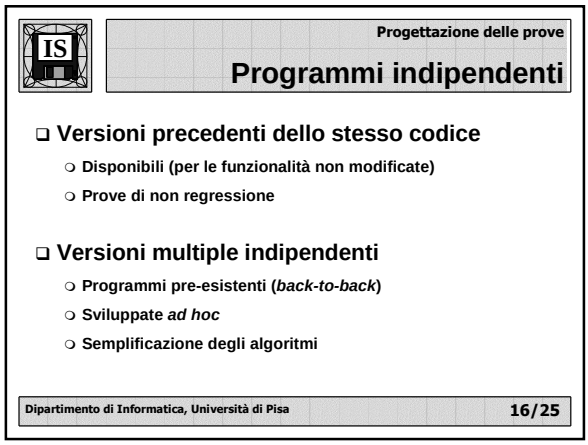

IS
Progettazione delle prove
Esempio di grafo di flusso
Funzione fattoriale
int fact(int n) {
if(n==0)
return 1;
else
return n*fact(n-1);
}
START
n == 0
n != 0
return 1
t ← n-1
k ← START(n ← t)
z ← n * k
return z
END
Dipartimento di Informatica, Università di Pisa
11/25


IS
Progettazione delle prove
Raffronto
Generalità degli approcci
Rispetto alla validità dei risultati
Rispetto alle caratteristiche da provare
Rispetto ai costi da sostenere
Dipendenze e implicazioni
L'applicazione di criteri funzionali non dipende dal codice
I criteri strutturali si prestano alla valutazione del grado di copertura
Dipartimento di Informatica, Università di Pisa
12/25

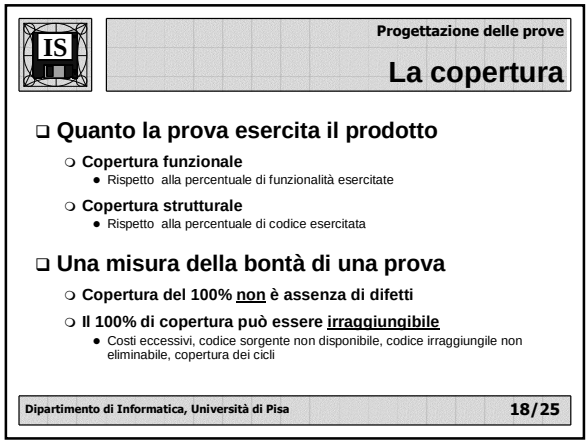

IS
Progettazione delle prove
L'oracolo
Metodo
Per generare i risultati attesi
Per validare i risultati (senza conoscerli)
Generalmente applicato da agenti automatici
Criteri generali
Basati sulle specifiche funzionali
Basati sulla semplificazione delle prove
Basati su componenti indipendenti
Dipartimento di Informatica, Università di Pisa
13/25

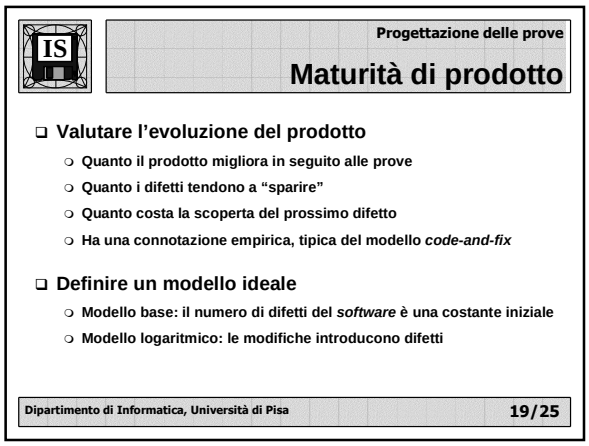

IS
Progettazione delle prove
Risultati dalle specifiche
Risultati ricavati dalle specifiche
Specifiche formali (generazione di invarianti)
Esempio: il numero di elementi di un vettore è invariante rispetto al suo ordinamento
Specifiche eseguibili (back-to-back)
Esempio: grafi causa-effetto
Inversione delle funzioni
Quando l'inversa è "più facile"
A volte disponibile fra le funzionalità di altri componenti
Limitazioni per difetti di approssimazione
Dipartimento di Informatica, Università di Pisa
14/25

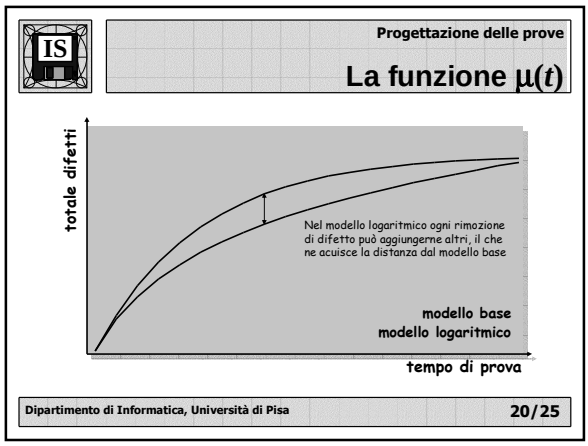

IS
Progettazione delle prove
Semplificazione dei dati
Semplificazione dei dati d'ingresso
Provare le funzionalità su dati "semplici"
Risultati noti o calcolabili con altri mezzi
Ipotesi di comportamento costante
Semplificazione dei risultati
Accontentarsi di risultati plausibili
Tramite vincoli fra ingressi e uscite
Tramite invarianti sulle uscite
Dipartimento di Informatica, Università di Pisa
15/25

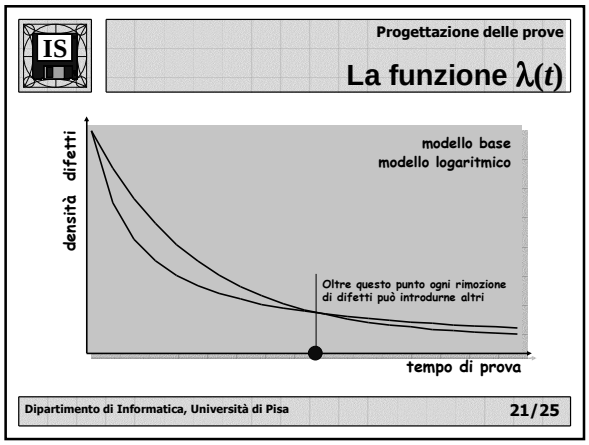

IS
Progettazione delle prove
Programmi indipendenti
Versioni precedenti dello stesso codice
Disponibili (per le funzionalità non modificate)
Prove di non regressione
Versioni multiple indipendenti
Programmi pre-esistenti (back-to-back)
Sviluppate ad hoc
Semplificazione degli algoritmi
Dipartimento di Informatica, Università di Pisa
16/25

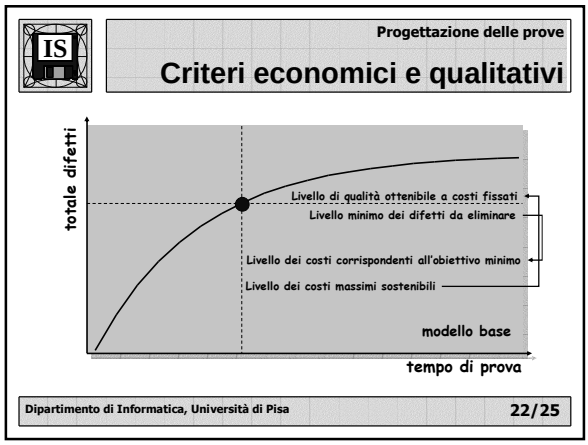

IS
Progettazione delle prove
Valutazione delle prove
Rapporto costi - confidenza
Le prove sono eseguite per verifica o per validazione
Devono fornire "adeguata confidenza"
Una prova fornisce solo certezza negativa (presenza di difetti)
La confidenza di una prova costa
Valutazione di una prova
Qualità di una prova in sé
Maggiore qualità → maggiore confidenza
Raggiungimento della confidenza desiderata
Dipartimento di Informatica, Università di Pisa
17/25

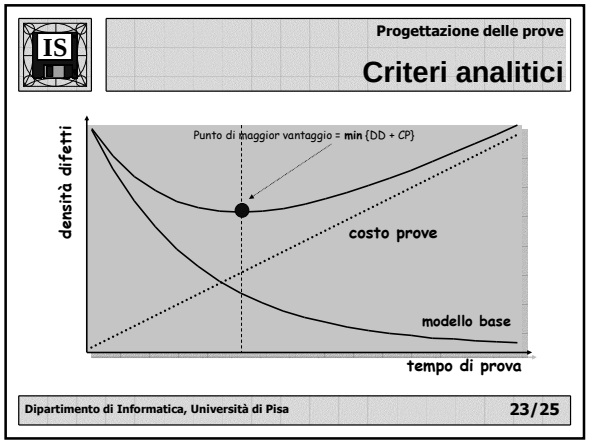

IS
Progettazione delle prove
La copertura
Quanto la prova esercita il prodotto
Copertura funzionale
Rispetto alla percentuale di funzionalità esercitate
Copertura strutturale
Rispetto alla percentuale di codice esercitata
Una misura della bontà di una prova
Copertura del 100% non è assenza di difetti
Il 100% di copertura può essere irraggiungibile
Costi eccessivi, codice sorgente non disponibile, codice irraggiungile non eliminabile, copertura dei cicli
Dipartimento di Informatica, Università di Pisa
18/25


IS
Progettazione delle prove
Maturità di prodotto
Valutare l'evoluzione del prodotto
Quanto il prodotto migliora in seguito alle prove
Quanto i difetti tendono a "sparire"
Quanto costa la scoperta del prossimo difetto
Ha una connotazione empirica, tipica del modello code-and-fix
Definire un modello ideale
Modello base: il numero di difetti del software è una costante iniziale
Modello logaritmico: le modifiche introducono difetti
Dipartimento di Informatica, Università di Pisa
19/25

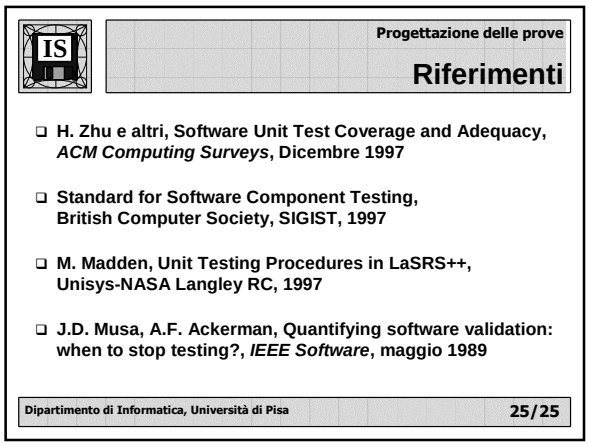

IS
Progettazione delle prove
La funzione μ(t)
totale difetti
Nel modello logaritmico ogni rimozione di difetto può aggiungerne altri, il che ne acuisce la distanza dal modello base
modello base
modello logaritmico
tempo di prova
Dipartimento di Informatica, Università di Pisa
20/25


IS
Progettazione delle prove
La funzione λ(t)
densità difetti
modello base
modello logaritmico
Oltre questo punto ogni rimozione di difetti può introdurne altri
tempo di prova
Dipartimento di Informatica, Università di Pisa
21/25


IS
Progettazione delle prove
Criteri economici e qualitativi
totale difetti
Livello di qualità ottenibile a costi fissati
Livello minimo dei difetti da eliminare
Livello dei costi corrispondenti all'obiettivo minimo
Livello dei costi massimi sostenibili
modello base
tempo di prova
Dipartimento di Informatica, Università di Pisa
22/25


IS
Progettazione delle prove
Criteri analitici
densità difetti
Punto di maggior vantaggio = min {DD + CP}
costo prove
modello base
tempo di prova
Dipartimento di Informatica, Università di Pisa
23/25


IS
Progettazione delle prove
Riepilogo
Caratteristiche delle prove del software
Criteri funzionali
Criteri strutturali
Analisi dei risultati, l'oracolo
Approfondimento: valutazione delle prove
Dipartimento di Informatica, Università di Pisa
24/25


IS
Progettazione delle prove
Riferimenti
H. Zhu e altri, Software Unit Test Coverage and Adequacy, ACM Computing Surveys, Dicembre 1997
Standard for Software Component Testing, British Computer Society, SIGIST, 1997
M. Madden, Unit Testing Procedures in LaSRS++, Unisys-NASA Langley RC, 1997
J.D. Musa, A.F. Ackerman, Quantifying software validation: when to stop testing?, IEEE Software, maggio 1989
Dipartimento di Informatica, Università di Pisa
25/25