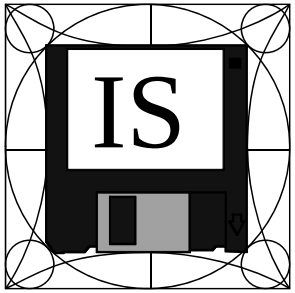
Verifica e validazione: introduzione



Ingegneria del Software

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

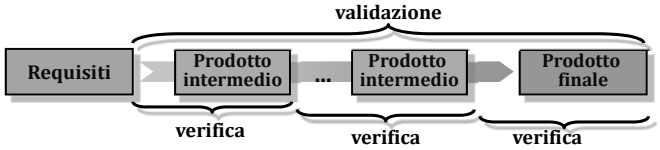
Aggiornamenti di: T. Vardanega (UniPD)

Definizione secondo ISO/IEC 12207

- *Software verification*
  - *Provides objective evidence that the outputs of a particular phase of the software development life cycle meet all of the specified [process] requirements for that phase*
  - *Software verification looks for consistency, completeness, and correctness [of the examined phase outputs], and provides support for a subsequent conclusion that software is validated*
- *Software validation*
  - *Confirmation by examination and provision of objective evidence that the software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled*

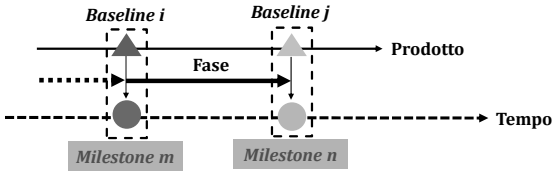
In pratica …

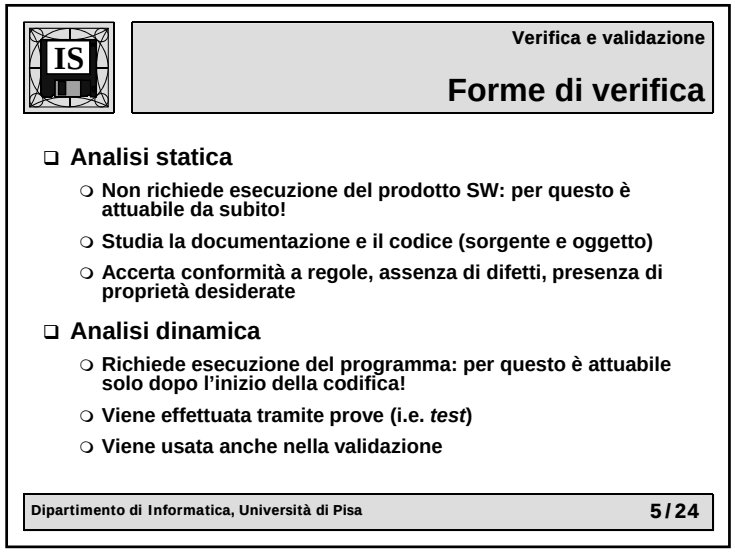
- La verifica accerta che l'esecuzione delle attività attuate nel periodo in esame («fase») non abbia introdotto errori
  - La verifica prepara il successo della validazione
- La validazione accerta che il prodotto realizzato sia pienamente conforme alle attese

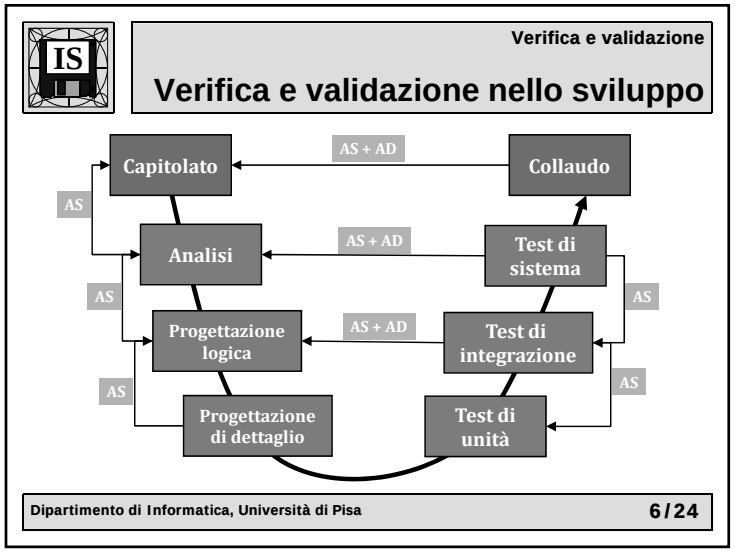


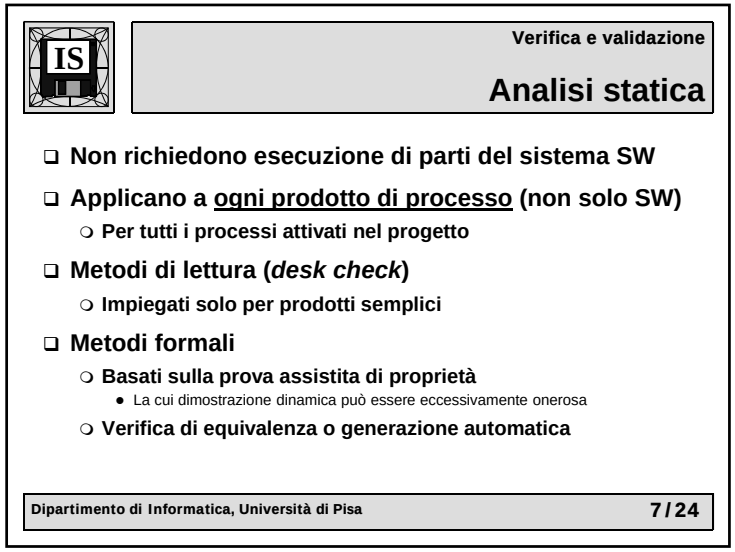
Quando svolgere verifica?

- A ogni incremento di *baseline*, che corrisponde concettualmente a una *milestone*
- Diciamo «fase» il tempo intercorrente tra due *milestone* successive




IS
Verifica e validazione
Forme di verifica
Analisi statica
Non richiede esecuzione del prodotto SW: per questo è attuabile da subito!
Studia la documentazione e il codice (sorgente e oggetto)
Accerta conformità a regole, assenza di difetti, presenza di proprietà desiderate
Analisi dinamica
Richiede esecuzione del programma: per questo è attuabile solo dopo l'inizio della codifica!
Viene effettuata tramite prove (i.e. test)
Viene usata anche nella validazione
Dipartimento di Informatica, Università di Pisa
5/24


IS
Verifica e validazione
Verifica e validazione nello sviluppo
Capitolato
AS + AD
Collaudo
AS
Analisi
AS + AD
Test di sistema
AS
AS
Progettazione logica
AS + AD
Test di integrazione
AS
AS
Progettazione di dettaglio
Test di unità
Dipartimento di Informatica, Università di Pisa
6/24


IS
Verifica e validazione
Analisi statica
Non richiedono esecuzione di parti del sistema SW
Applicano a ogni prodotto di processo (non solo SW)
Per tutti i processi attivati nel progetto
Metodi di lettura (desk check)
Impiegati solo per prodotti semplici
Metodi formali
Basati sulla prova assistita di proprietà
La cui dimostrazione dinamica può essere eccessivamente onerosa
Verifica di equivalenza o generazione automatica
Dipartimento di Informatica, Università di Pisa
7/24


IS
Verifica e validazione
Metodi di lettura
Walkthrough e Inspection
Svolte tramite studio dell'oggetto di verifica
Lettura umana o automatizzata
Con efficacia dipendente dall'esperienza dei verificatori
Nell'organizzare le attività da svolgere
Nel documentare le risultanze
Modalità relativamente complementari
Dipartimento di Informatica, Università di Pisa
8/24

Walkthrough: definizione

- **Obiettivo**
  - **Rilevare la presenza di difetti attraverso lettura critica del prodotto in esame**
    - Analisi a largo spettro senza assunzione di presupposti
- **Agenti**
  - **Gruppi misti verificatori/sviluppatori, ma ruoli distinti**
- **Strategia**
  - **Per il codice: percorrerlo simulandone possibili esecuzioni**
  - **Per documenti: studiarne ogni parte come farebbe un compilatore di programmi**

Walkthrough: attività

- **Fase 1: pianificazione (congiunta)**
- **Fase 2: lettura (verificatori)**
- **Fase 3: discussione (congiunta)**
- **Fase 4: correzione dei difetti (sviluppatori)**
- **In ogni fase, documentazione delle attività svolte e delle risultanze**

Inspection: definizione

- **Obiettivi**
  - **Rilevare la presenza di difetti, eseguendo lettura mirata dell'oggetto di verifica**
- **Agenti**
  - **Verificatori**
- **Strategia**
  - **Focalizzare la ricerca su presupposti, tramite *error guessing***

Inspection: attività

- **Fase 1: pianificazione**
- **Fase 2: definizione lista di controllo**
  - **Specifica cosa verificare selettivamente**
- **Fase 3: lettura**
- **Fase 4: correzione dei difetti (a carico degli sviluppatori)**
- **In ogni fase, documentazione delle attività svolte e delle risultanze**

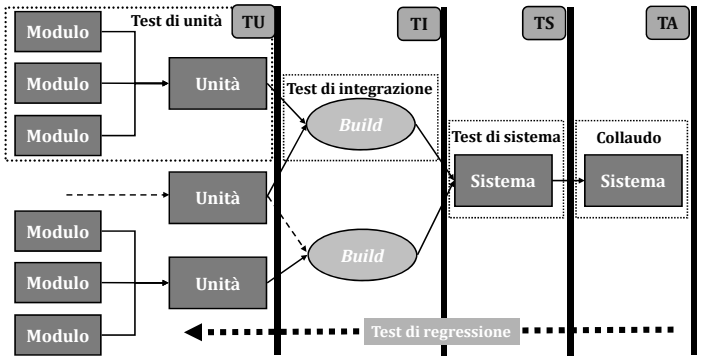
Inspection vs. *walkthrough*

- **Affinità**
  - **Entrambe basate su *desk check***
  - **Verificatori e sviluppatori su fronti opposti**
  - **Documentazione rigorosa di attività ed esiti**
- **Differenze**
  - ***Inspection* basato su presupposti**
  - ***Walkthrough* richiede maggiore attenzione**
  - ***Walkthrough* più collaborativo**
  - ***Inspection* più rapido**

Analisi dinamica: ambiente di prova

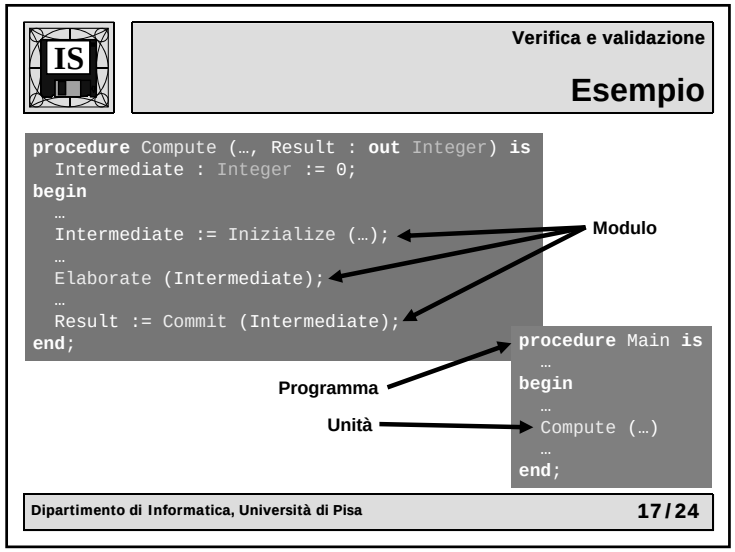
- **I *test* devono essere ripetibili: per questo specificano**
  - **Ambiente d'esecuzione: HW/SW, stato iniziale**
  - **Attese: ingressi richiesti, uscite ed effetti attesi**
  - **Procedure: esecuzione, analisi dei risultati**
- **I *test* vanno automatizzati: per questo usano strumenti**
  - ***Driver*** **componente attiva fittizia per pilotare il *test***
  - ***Stub*** **componente passiva fittizia per simulare la parte del sistema utile al *test* ma non oggetto di esso (vedere alla voce: «Per approfondire #22»)**
  - ***Logger*** **componente non intrusivo di registrazione dei dati di esecuzione per analisi dei risultati**

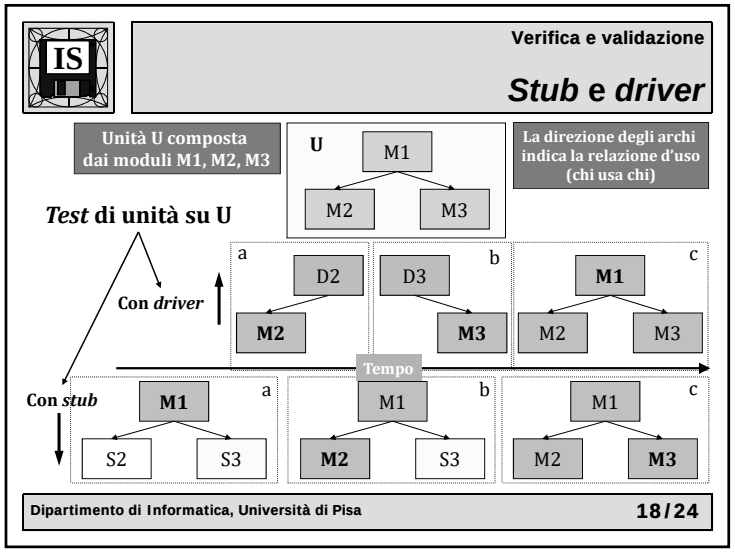
Analisi dinamica: tipi di *test*



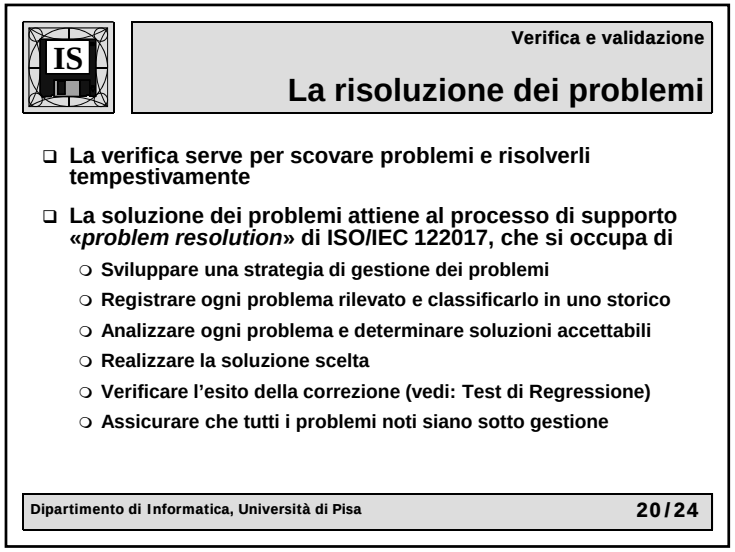
Glossario

- **Unità**
  - **La più piccola quantità di SW che sia utilmente sottoponibile a verificare individuale**
  - **Tipicamente prodotta da un singolo programmatore**
  - **Va intesa in senso architetturale: non linee di codice ma entità di organizzazione logica**
    - Singola procedura, singola classe, piccolo aggregato (*package*)
- **Il modulo (come determinato dal linguaggio di programmazione) è una frazione dell'unità**
- **Il componente integra più unità correlate e coese**


Verifica e validazione
Esempio
procedure Compute (…, Result : out Integer) is
Intermediate : Integer := 0;
begin
…
Intermediate := Inizialize (…);
…
Elaborate (Intermediate);
…
Result := Commit (Intermediate);
end;
Modulo
Programma
Unità
procedure Main is
…
begin
…
Compute (…)
…
end;
Dipartimento di Informatica, Università di Pisa
17/24


Verifica e validazione
Stub e driver
Unità U composta dai moduli M1, M2, M3
U
M1
M2
M3
La direzione degli archi indica la relazione d'uso (chi usa chi)
Test di unità su U
Con driver
a
D2
M2
b
D3
M3
c
M1
M2
M3
Tempo
Con stub
M1
S2
S3
M1
M2
S3
M1
M2
M3
Dipartimento di Informatica, Università di Pisa
18/24


Verifica e validazione
Test di unità
È agevolata da attività mirate di analisi statica
Per determinare limiti di iterazioni, flusso di programma, valori di variabili, ecc.
Si svolge con il massimo grado di parallelismo e desiderabilmente con automazione
Poiché i TU sono tanti
È sotto la responsabilità dello stesso programmatore per le unità più semplici
Di un verificatore indipendente altrimenti
Deve accertare la correttezza del codice as implemented
Dipartimento di Informatica, Università di Pisa
19/24


Verifica e validazione
La risoluzione dei problemi
La verifica serve per scovare problemi e risolverli tempestivamente
La soluzione dei problemi attiene al processo di supporto «problem resolution» di ISO/IEC 122017, che si occupa di
Sviluppare una strategia di gestione dei problemi
Registrare ogni problema rilevato e classificarlo in uno storico
Analizzare ogni problema e determinare soluzioni accettabili
Realizzare la soluzione scelta
Verificare l'esito della correzione (vedi: Test di Regressione)
Assicurare che tutti i problemi noti siano sotto gestione
Dipartimento di Informatica, Università di Pisa
20/24

Test di regressione

- Modifiche effettuate per aggiunta, correzione o rimozione, non devono pregiudicare le funzionalità già verificate, causando regressione
  - Il rischio aumenta all'aumentare dell'accoppiamento e al diminuire dell'incapsulazione
- Il *test* di regressione comprende tutti i *test* necessari ad accertare che la modifica di una parte P di S non causi errori in P, in S, o in ogni altra parte del sistema che sia in relazione con S
  - Meglio se ripetendo *test* già specificati ed eseguiti

Test di integrazione

- Per costruzione e verifica incrementale del sistema
  - Componenti sviluppati in parallelo e verificati incrementalmente
  - La *build* incrementale è automatizzabile
  - In condizioni ottimali l'integrazione è priva di problemi
- Quali problemi rileva
  - Errori residui nella realizzazione dei componenti
  - Modifica delle interfacce o cambiamenti nei requisiti
  - Riuso di componenti dal comportamento oscuro o inadatto
  - Integrazione con altre applicazioni non ben conosciute

Test di sistema e collaudo

- Validazione
  - *Test* di sistema come attività interna del fornitore
    - Per accertare la copertura dei requisiti SW
  - Collaudo come attività supervisionata dal committente
    - Per dimostrazione di conformità del prodotto sulla base di casi di prova specificati nel o implicati dal contratto
- Implicazioni contrattuali
  - Il collaudo è attività formale alla presenza del committente
  - Il rilascio del prodotto (e l'eventuale fine del progetto) segue al collaudo andato a buon fine

Riferimenti

- Standard for Software Component Testing, British Computer Society SIGIST, 1997
- M.E. Fagan, Advances in Software Inspection, *IEEE Transaction on Software Engineering*, luglio 1986
- G.A. Cignoni, P. De Risi, "Il test e la qualità del software", Il Sole 24 Ore, 1998