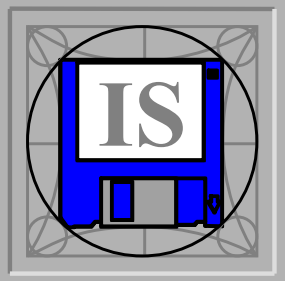
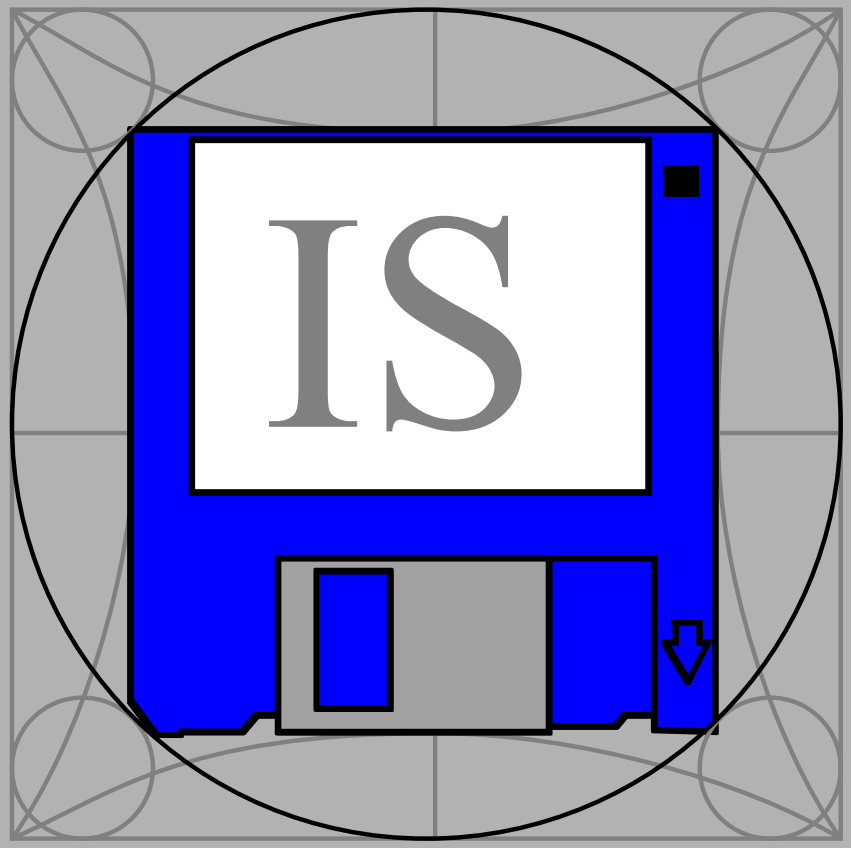


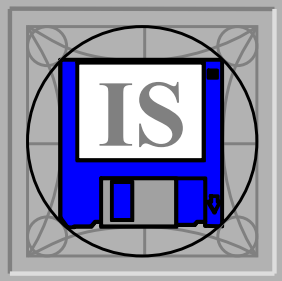
Verifica e validazione: introduzione



Ingegneria del Software

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

Aggiornamenti di: T. Vardanega (UniPD)



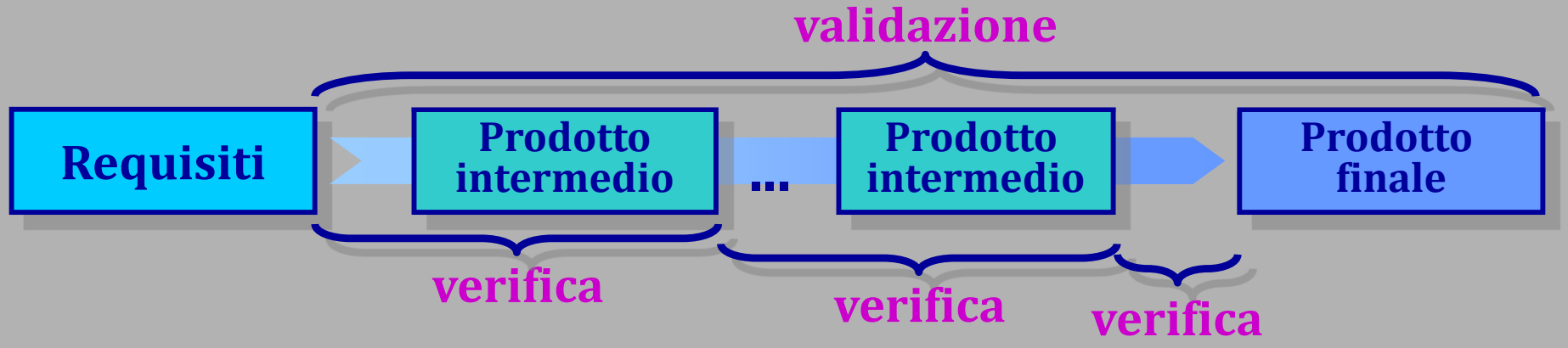
Secondo ISO/IEC 12207

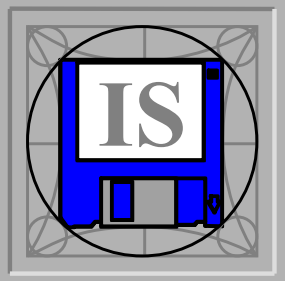
- ***Software verification***
  - *Provides objective evidence that the outputs of a particular segment of the software development meet all the requirements specified for it*
  - *Looks for consistency, completeness, and correctness of those outputs*
  - *Provides support for subsequent conclusion that software is validated*
- ***Software validation***
  - *Confirmation by examination and provision of objective evidence that*

1. *the SW specifications conform to user needs and intended uses*
2. *the requirements implemented through SW can be consistently fulfilled*

In pratica …

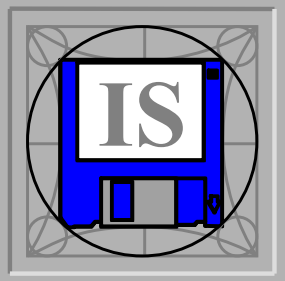
- La **verifica** agisce su singoli segmenti di sviluppo, accertando che l'esecuzione in essi (segmenti) non abbia introdotto errori
  - Approvando le *baseline* associate alle *milestone* di progetto
- La **validazione** agisce a fine progetto, accertando che il prodotto finale sia pienamente conforme alle aspettative
- La verifica prepara il successo della validazione





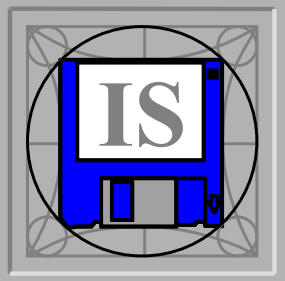
Ripassiamo quel che già sappiamo

- Una *milestone* è una data di calendario che fissa un punto di avanzamento atteso
- Il raggiungimento di quegli obiettivi di avanzamento va sostanziato da una *baseline*
- Una *baseline* è la versione approvata di un prodotto di lavoro (parte di un progetto) che può essere modificato solo attraverso procedure formali di controllo delle modifiche
- Il prodotto di progetto è un aggregato di SW e di documentazione



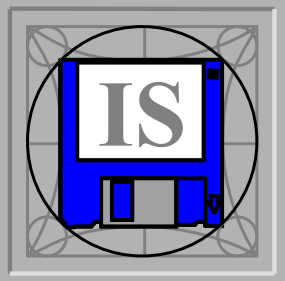
La verifica ha due forme

- **Analisi statica (AS)**
  - Non richiede esecuzione dell'oggetto di verifica
  - Studia documentazione e codice (sorgente, oggetto)
  - Accerta conformità a regole, assenza di difetti, presenza di proprietà desiderate
- **Analisi dinamica (AD)**
  - Richiede esecuzione dell'oggetto di verifica (*SW*)
  - Viene effettuata tramite prove (*test*)
  - Viene usata anche nella validazione



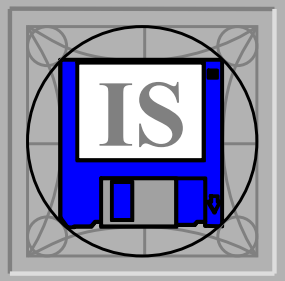
Analisi statica

- Non richiedendo esecuzione dell'oggetto di verifica, è applicabile a *ogni* prodotto di processo
  - Per tutti i processi attivati nel progetto
- Può usare metodi di lettura (*desk check*)
  - Impiegati solo per prodotti semplici
- Oppure metodi più formali
  - Basati su prova assistita di proprietà, utile soprattutto quando la dimostrazione empirica ha costo proibitivo
  - (Di questi parliamo nella lezione successiva)



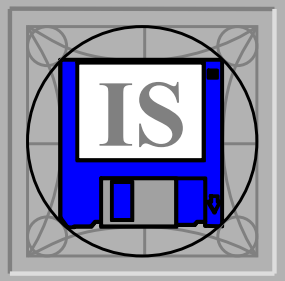
Metodi di lettura

- *Walkthrough* e *Inspection*
  - Svolte tramite lettura dell'oggetto di verifica
  - Lettura umana o automatizzata
- La loro efficacia dipende dall'esperienza dei verificatori
  - Nell'organizzare le attività da svolgere
  - Nel documentare le risultanze
- Modalità relativamente complementari tra loro



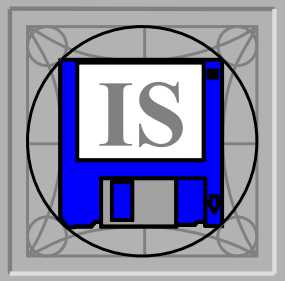
Walkthrough: definizione

- **Obiettivo**
  - Rilevare la presenza di difetti attraverso lettura critica ad ampio spettro dell'oggetto in esame
- **Agenti**
  - Gruppi misti autori / sviluppatori, con ruoli distinti tra loro
- **Strategia**
  - Esame privo di assunzioni o presupposti
- **Modalità**
  - Percorrere il codice simulandone possibili esecuzioni
  - Studiare ogni parte di documento, come farebbe un compilatore



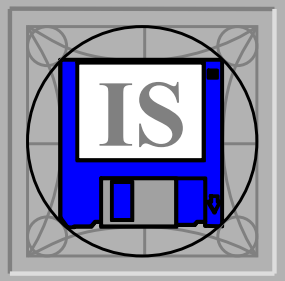
Walkthrough: attività

- **Passo 1: pianificazione**
  - Autori e verificatori
- **Passo 2: lettura**
  - Solo verificatori
- **Passo 3: discussione**
  - Autori e verificatori
- **Passo 4: correzione dei difetti**
  - Solo autori
- **Ogni passo documenta attività svolte e risultanze**



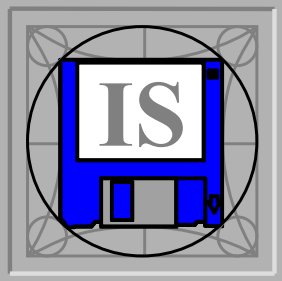
Inspection: definizione

- **Obiettivi**
  - Rilevare la presenza di difetti eseguendo lettura mirata dell'oggetto di verifica
- **Agenti**
  - Verificatori
- **Strategia**
  - Ricerca focalizzata su presupposti (*error guessing*)
- **Modalità**
  - Sapendo cosa cercare permette automazione della ricerca



Inspection: attività

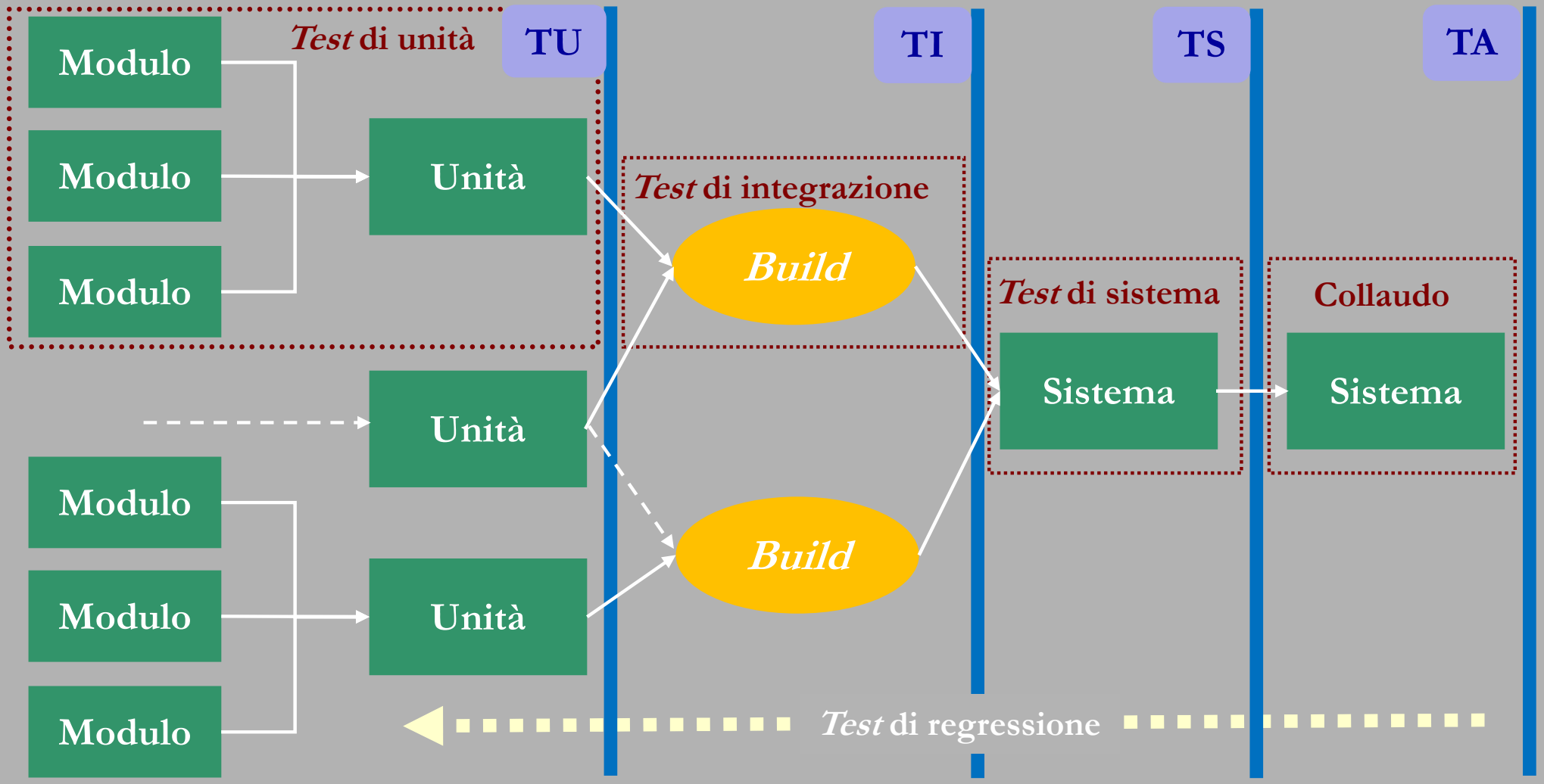
- Passo 1: pianificazione
- Passo 2: definizione lista di controllo
  - Cosa vada verificato selettivamente
- Passo 3: lettura
- Passo 4: correzione dei difetti
  - A carico degli autori
- Ogni passo documenta attività svolte e risultanze

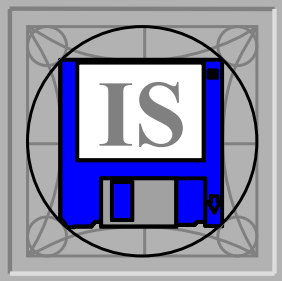


Analisi dinamica: ambiente di prova

- 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: perciò usano strumentazione
  - *Driver* componente attiva fittizia per pilotare il *test*
  - *Stub* componente passiva fittizia per simulare parti del sistema utili al *test* ma non oggetto di *test*
  - *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 verifica in isolamento
  - 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

Esempio

```
procedure Main is
   …
begin
   …
   Compute (…)
   …
end;
```

Programma

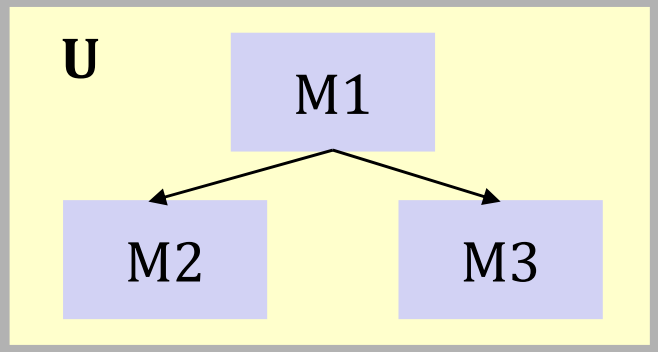
Unità

```
procedure Compute (…, Result : out Integer) is
   Intermediate : Integer := 0;
begin
   …
   Intermediate := Initialize (…);
   …
   Elaborate (Intermediate);
   …
   Result := Commit (Intermediate);
end;
```

Modulo

Stub e *driver* – 2/2

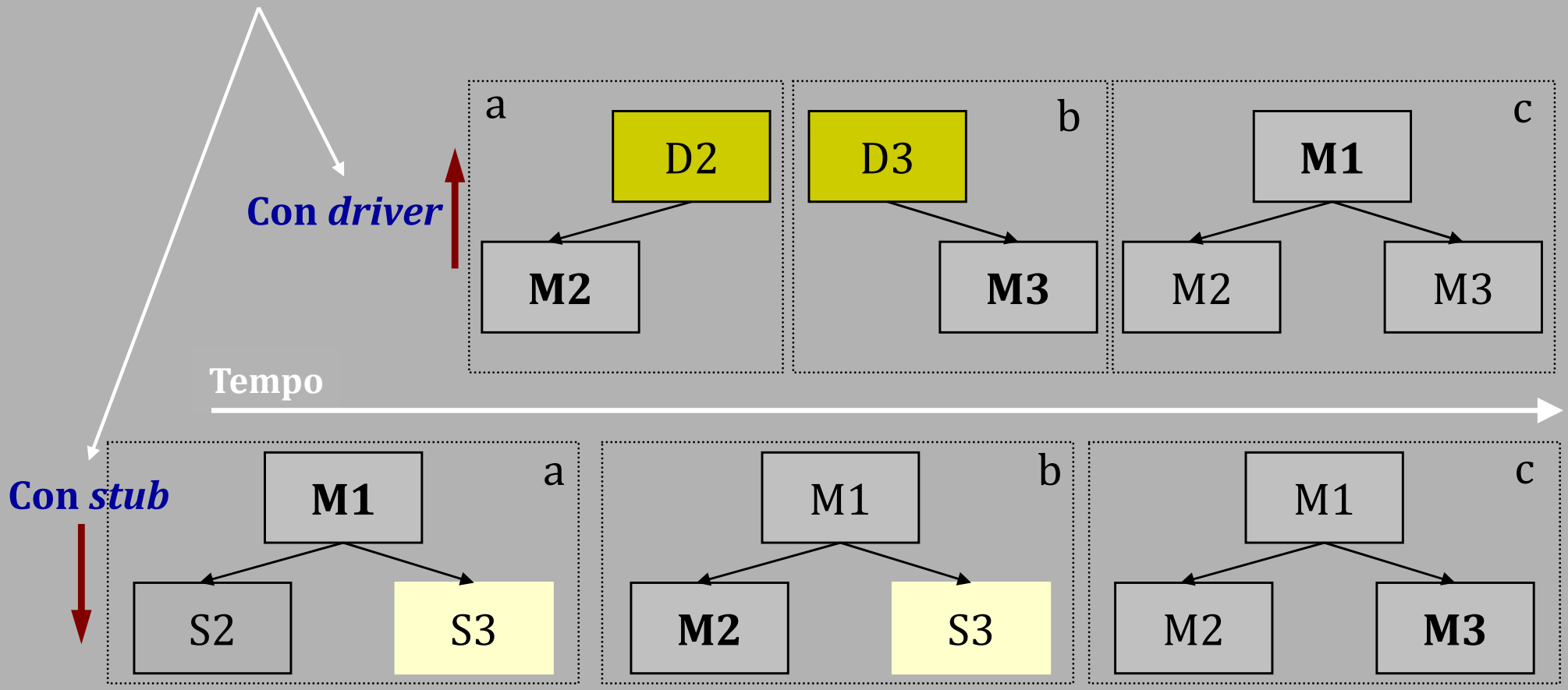
Unità U composta dai moduli M1, M2, M3

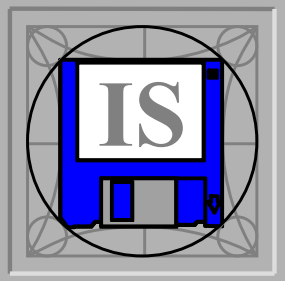


La direzione degli archi indica la relazione d'uso (chi usa chi)

Stub e *driver* – 2/2

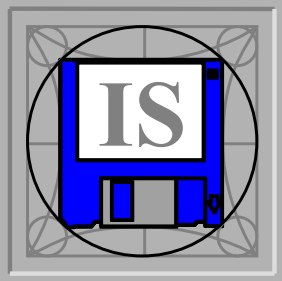
Test di unità su U





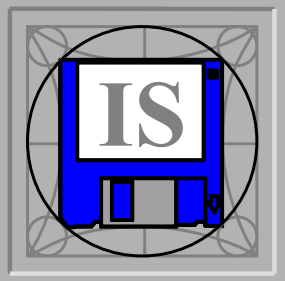
Test di unità

- È agevolato da attività mirate di analisi statica
  - Limiti di iterazioni, flusso di esecuzione, valori di variabili, ...
- Consente alto grado di parallelismo e automazione nell'esecuzione
- Per le unità più semplici, può essere a carico del loro stesso autore
  - Altrimenti meglio assegnarlo a verificatore indipendente
- Accerta la correttezza del codice «*as implemented*»
  - Mai modificare il sorgente del codice cui si esegue *test*



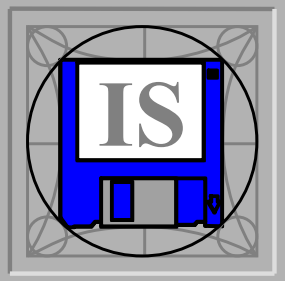
La risoluzione dei problemi

- 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
  - Assicurare che tutti i problemi noti siano sotto gestione



Test di regressione

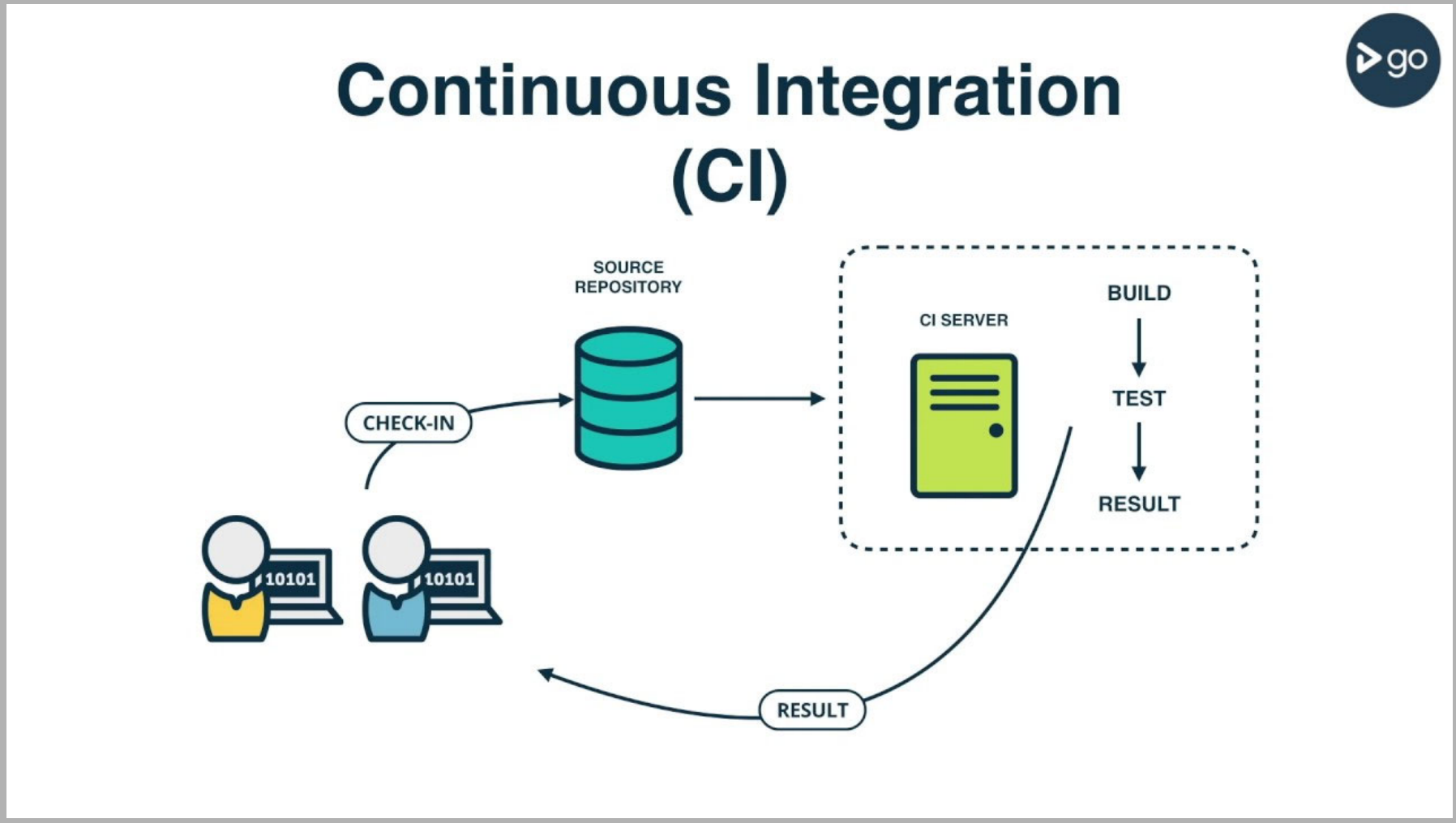
- Modifiche effettuate per aggiunta, correzione o rimozione, non devono pregiudicare le funzionalità già verificate,
- Se lo fanno, causano regressione
  - Il rischio di regressione aumenta all'aumentare dell'accoppiamento e al diminuire dell'incapsulazione
- Il *test* di regressione comprende tutti 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
  - Ripetendo *test* già specificati e già eseguiti

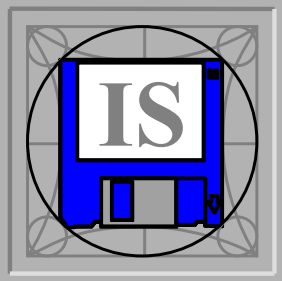


Test di integrazione – 1/2

- Per costruzione e verifica incrementale del sistema
  - Quando l'integrazione incrementale di componenti sviluppati in parallelo realizza funzionalità di livello sistema
  - La *build* incrementale è totalmente 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 integrazione – 2/2

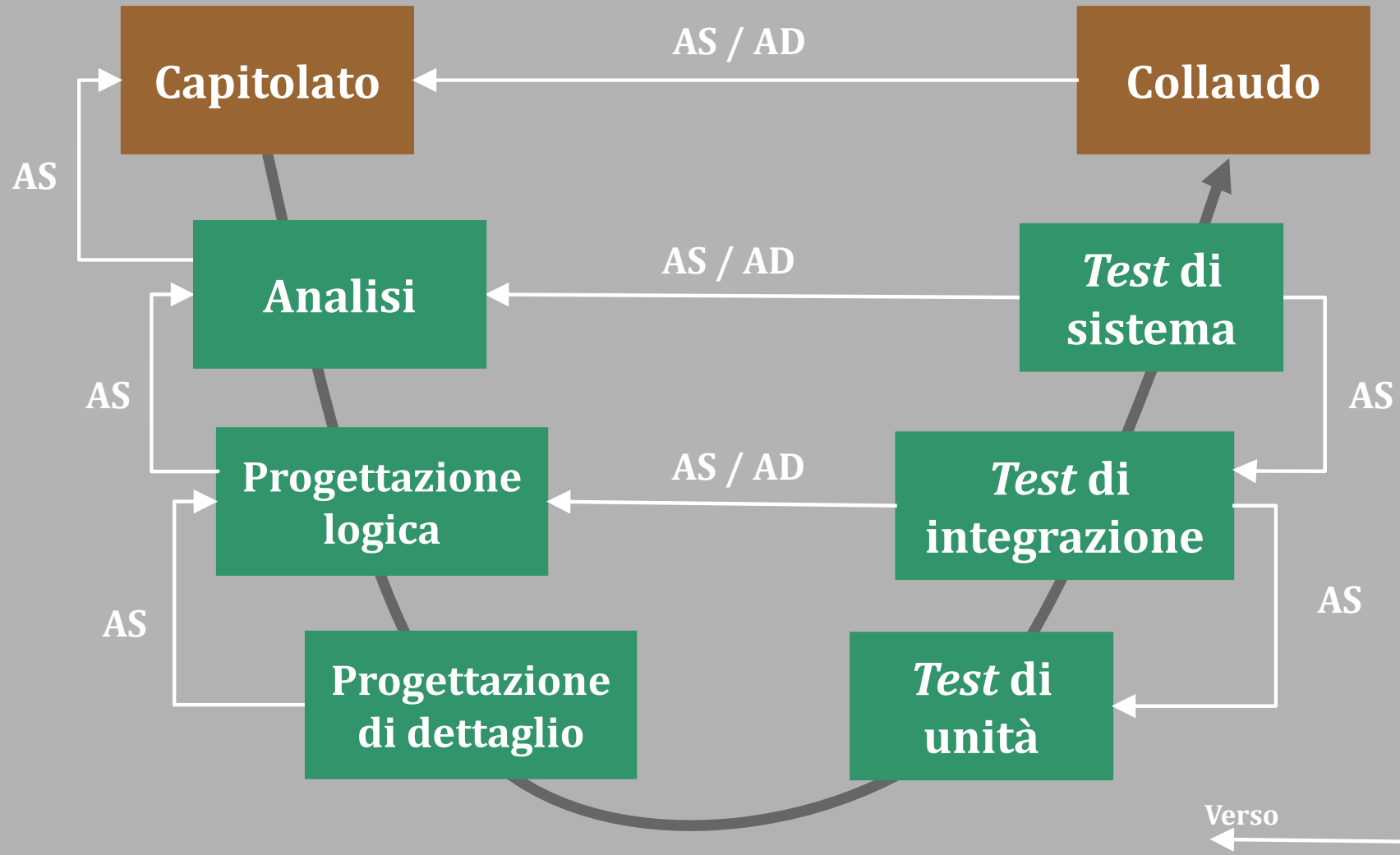


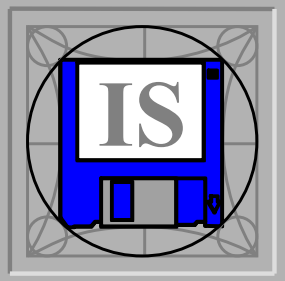


Test di sistema e collaudo

- **Validazione**
  - ***Test* di sistema come attività interna del fornitore**
    - Per accertare la copertura dei requisiti SW in preparazione al collaudo
  - **Collaudo come attività supervisionata dal committente**
    - Per dimostrare conformità del prodotto alle attese attraverso casi di prova implicati dal capitolato
- **Collaudo**
  - **Attività formale di fronte al committente**
  - **Al suo buon esito consegue rilascio finale del prodotto**

Visione d'insieme





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