Verifica e validazione(12)

- **Verifica:** "ho fatto il sistema nel modo giusto", accerta che l'esecuzione delle attività di processi svolti nella fase in esame non abbia introdotto errori nel prodotto;
- **Validazione:** "ho fatto il sistema giusto", accerta che il prodotto realizzato sia conforme alle attese.

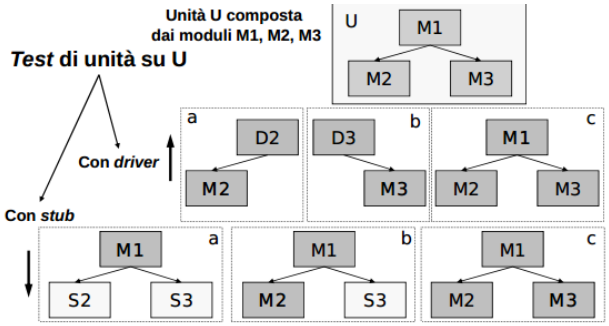
La *software verification* ricerca la completezza e la correttezza del software e tratta ciò che lo supporta. Consente di valutare di conseguenza che il sw sia validato. La verifica è a supporto della validazione e la validazione è l'ultima cosa che faccio in un progetto. La verifica è un attività che svolgo durante **tutto lo sviluppo** fino all'ultimo istante dove farò validazione, che servirà a dire che ciò che ho fatto è la cosa giusta. La verifica va fatta per impedire che la risposta finale non sia sbagliata. Devo garantire tre cose importanti:

- **Consistenza:** "sono ciò che vi attendevate fossi";
- **Correttezza:** "ciò che ho conseguito è corretto rispetto alle norme";
- **Completezza:** "tutto ciò che ho creato è tutto ciò era atteso".

Sono tre caratteristiche di cui devo accertare l'esistenza su tutti i prodotti parziali dello sviluppo. Il verificatore impara le norme e dice che quello che è stato fornito è fatto come richiesto. Non corregge nè rifà il lavoro ma controlla solo che tutto rispetti le tre caratteristiche. La validazione conseguentemente è una conferma **by examination**, mostra copertura dei requisiti (utente e sw).
Per il verificatore ho due forme di **analisi**:

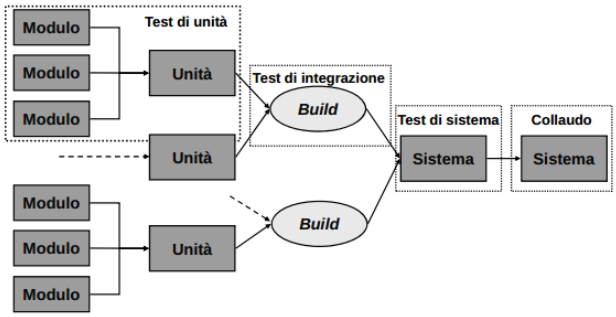
- **Analisi statica:** non richiede l'esecuzione del programma, studia le caratteristiche del codice sorgente (e a volte anche del codice oggetto), conformità a regole date, assenza di difetti, presenza di proprietà positive;
- **Analisi dinamica:** richiede l'esecuzione del programma, viene effettuata tramite **test**, usata sia nella verifica che nella validazione.
  - **Ripetibilità:** è un requisito essenziale. Dobbiamo assumere uno stato iniziale prima dell'esecuzione in quanto ha influenza sia diretta che indiretta sull'esecuzione. Il test deve essere **deterministico** ed eseguire le cose secondo un ordine noto. **Specifica di un test**;
  - **Strumenti**:
    - **Driver:** componente attiva fittizia per pilotare una parte;
    - **Stub:** componente passiva fittizia per simulare una parte;
    - **Logger:** componente non intrusiva di registrazione dei dati di esecuzione per l'analisi dei risultati. Ogni tanto deve lasciare traccia del suo esito;

  item **Unità:** può essere anche un aggregato di procedure. La più piccola unità sw che è conveniente verificare singolarmente. Un *modulo* è parte dell'unità, un *componente* integra più unità.



Con **stub** ho dei test *Top down* dalla radice alle foglie, con **driver** ho dei test *Bottom up* dalle foglie alla radice

Tipi di test



- **Test di unità:** si svolgono con il massimo grado di parallelismo, la responsabilità è dello stesso programmatore sulle unità più piccole. L'obbiettivo è quello di verificare la correttezza del codice.
- **Test di integrazione:** le componenti vengono verificate e sviluppate in parallelo, rileva errori residui nella realizzazione dei componenti, cambiamenti nelle interfacce, requisiti, integrazione con altre applicazioni non ben conosciute.
- **Test di sistema e collaudo:** dai requisiti so dire quanti test di sistema avrò. I test di sistema è un'attività interna del fornitore per accertare la copertura dei requisiti, il collaudo invece viene supervisionato dal committente.
- **Test di regressione:** è l'insieme dei test che accertano che la modifica di una parte P non causi problemi in P o in altre parte che dipendono da essa, infatti modifiche aggiunte o rimozioni non devono pregiudicare le funzionalità già verificate.

Analisi Statica

Si può applicare ai metodi di lettura che si possono suddividere in due tipo:

- **Walkthrought:** l'obbiettivo è quello di rilevare la presenza di difetti, si esegue una lettura a largo spetto senza l'assunzione di presupposti, le fasi sono: pianificazione, lettura, discussione, correzione dei difetti.
- **Inspection:** l'obbiettivo è sempre quello di rilevare difetti ma eseguendo una lettura mirata, si focalizza la ricerca su presupposti, le fasi sono: pianificazione, definizione della lista di controllo, lettura, correzione dei difetti.

Inspection è basato su errori presupposti ed è più rapido, *Walkthrought* richiede maggiore attenzione ma è più collaborativo.

Valori dell'**Analisi Statica**:

- **Funzionalità:** analisi statica come attività preliminare, liste di controllo rispetto ai relativi requisiti *(tutte e solo le funzionalità per tutti e solo i componenti necessari, compatibilità tra tutte le soluzioni adottate)*, valutazione di accuratezza;
- **Affidabilità:** dimostrabile tramite combinazione di prove, analisi statica come attività preliminare, liste di controllo rispetto ai relativi requisiti *(robustezza, capacità di ripristino e recupero da errori, adesione alle norme)*, valutazione di maturità.

- **Usabilità:** le prove sono imprescindibili, analisi statica come attività complementare, liste di controllo rispetto ai manuali d'uso *(comprensibilità, apprendibilità, adesione a norme e prescrizioni)*, questionari sottomessi agli utenti.
- **Efficenza:** le prove sono necessarie, analisi statica come attività complementare, liste di controllo rispetto alle norme di codifica, margini di miglioramento e confidenza grazie alla confidenza acquisita.
- **Manutenibità:** analisi statica come strumento ideale, liste di controllo rispetto a specifiche norme di codifica, e alla prove per accertarne, prove di stabilità.
- **Portabilità:** analisi statica come strumento ideale, liste di controllo rispetto a specifiche norme di codifica, prove come strumento complementare.