Verifica e Validazione

H1 # Introduzione

Validazione

H2
> Verifica che il SW sia **corretto**:
>
> H5
> - accertamento che un SW soddisfi i requisiti dell'utente
> - *building the right system*

Verifica

> Verifica che l'implementazione corrisponda alla sua specifica:
>
> H5
> - *building the system right*

V&V

H3 ## V&V del progetto

H2

La **verifica e validazione a livello di progetto** è molto importante per limitare errori che risultano costosi più in là nello sviluppo. Per farla si può:

- Effettuare una **revisione** del progetto prima dell'implementazione, indipendentemente da quanto il processo di sviluppo sia formale
- Usare **revisori esterni**, cioè altre persone non coinvolte direttamente che potrebbero notare errori
- Usare **checklist** contenente una lista di controlli
- Uso di **metodi formali**, per dimostrare la correttezza del progetto

Tecniche per l'analisi di specifiche formali

Scrivere le specifiche in modo formale permette l'uso di tecniche quali:

H4

- **Analisi sintattiche**:
  - **Completezza:**
    ogni evento che può avvenire viene descritto
  - **Consistenza**:
    la reazione a ogni evento è descritta in modo non ambiguo e univoco
- **Verifica formale di proprietà**:
  dimostrazione formale che le proprietà desiderate siano implicate dal modello
- **Simulazione e esecuzione scenari**:
  simulazione di scenari o casi d'uso per essere sicuri che le specifiche siano aderenti ai requisiti dell'utente

H3 V&V dell'implementazione

Le tecniche di **V&V** a livello di implementazione si dividono in:

- ***Statiche:*** analisi del codice senza eseguirlo
  - revisioni, **ispezioni**, etc.
  - **analisi statica** automatica del codice
  - **verifica formale** di correttezza
- ***Dinamiche:*** analisi del codice eseguendolo
  - **testing**
  - *profiling:* valutazione delle prestazioni

Analisi statica del codice

SI tratta di analizzare il **flusso di controllo** e il **flusso dei dati**. Solitamente, ciò viene fatto dal compilatore. Questo tipo di analisi evita errori quali:

H4

- istruzioni non raggiungibili
- variabili usate prima di essere inizializzate
- variabili dichiarate e mai usate
- variabile scritte e mai lette

- variabili mai usate tra due assegnamenti
- possibili violazioni di array
- . . .

Verifica formale

Un programma si può *dimostrare* **corretto** usando la *logica*.

H4 La logica più utilizzata è la **logica di Hoare**, che permette di esprimere la specifica mediante precondizioni e postcondizioni.

Ispezione (di *Fagan*)

L'**ispezione** è una tecnica manuale che prevede l'analisi del codice per trovare errori. Il modello di *Fagan* è fra quelli più usati perché abbastanza
H4 strutturato.

Nel suo modello abbiamo i seguenti ruoli:

- **Autore:** chi ha scritto il sorgente
  - partecipante passivo, risponde a domande quando richiesto
- **Lettori:**
  - leggono il codice al gruppo, cercano difetti, possono usare delle **checklist**
- **Moderatore:** tipicamente proviene da un altro progetto
  - presiede le sedute, sceglie i partecipanti, controlla il processo di ispezione

Considerazioni

I difetti trovati durante l'**ispezione** non devono essere usati per la valutazione personale. Il programmatore così non ha motivo di
H5 nascondere gli errori.

Invece, i difetti trovati durante il **testing**, cioè dopo l'ispezione, sono usati per la valutazione personale. I programmatori sono così incentivati a trovare difetti durante l'ispezione.

Vantaggi

✔ processo **formale**, con tracciamento dei risultati

✔ aspetti **sociali** del processo

H6

Svantaggi

✕ **Non scalabile**, si limita alla verifica di unità

H6

Analisi dinamica

Esistono diversi tool, quali:

H4

- **Runtime checkers:**
  analizzano l'esecuzione del codice per vedere se ci sono comportamenti errati (violazioni di memoria, deadlock, etc.), prima di passare le chiamate al OS.
- **Penetration testing tools:**
  controllano l'applicazione secondo schemi di vulnerabilità conosciuti