# 12 pt5 Static analisis: 298

Analisi statica del codice alla ricerca di malfunzionamenti (ispezione + analisi source code):

* **Vedere problemi di compilazione**
  + Controlli semantici (sintassi+tipo)
  + Molto dipendente dal linguaggio che si usa
  + Es: Misra-C → verificano che codice siano in line con lo standard misra
    - Vari esempi su slide
* Analisi del **control flow**
* **Data flow**
  + Analizza i valori di una variabile durante un’esecuzione per cercare anomalie
    - Definizione e scrittura nuovo valore
    - Uso, lettura del valore della variabile
    - Immagine che contiene testo, schermata, Carattere, linea

      Descrizione generata automaticamenteNullification → valore senza significato

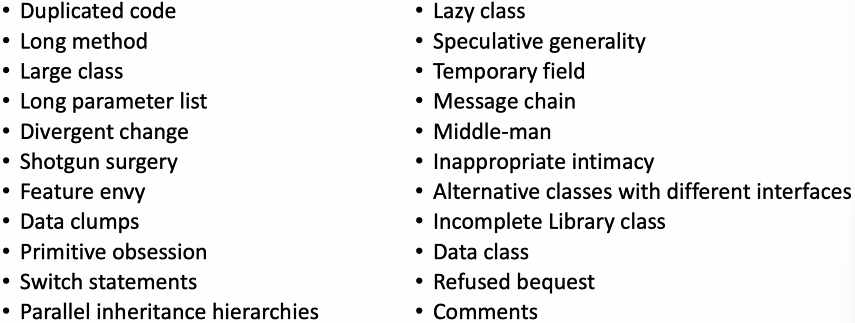
Nota: sequenze corrette sono D- U *(uso di una variabile deve essere sempre preceduto da una definizione della stessa variabile)*

Nota: sequenze sospette: D-U / N-U *(L'uso di una variabile non preceduta da una definizione corrisponde all'uso di un valore non definito)*

* **Symbolic execution**
  + Immagine che contiene testo, schermata, Carattere, numero

    Descrizione generata automaticamenteProgramma eseguito con valori simbolici
* Cercare di derivare il design del sw a partire dalla static analysys
  + **Reverse documentation of design/code**
    - Immagine che contiene testo, Carattere, linea, schermata

      Descrizione generata automaticamenteDurante il tempo, il codice cambia e può non corrispondere al design inziiale

**Nota**: In compilazione potremmo trovare i **Bad Smells**: istruzioni che se create, possono portare malfunzionamenti

*EsLint, SonarQube, CodeClimate svolgono questo tipo di lavoro*

***VandV si occupa di prevenire, trovare e correggere i difetti***

* *Molte tecniche disponibili: test, ispezione, analisi statica*
* *Il costo e l'impegno di VandV dovrebbero essere confrontati con il costo dei difetti lasciati all'utente finale* 
  + *Costo della qualità vs. costo della non qualità*