SOLID

- **S** - Single-responsiblity principle
- **O** - Open-closed principle
- **L** - Liskov substitution principle
- **I** - Interface segregation principle
- **D** - Dependency Inversion Principle

O clasă ar trebui să aibă un singur motiv pentru a se schimba, ceea ce înseamnă că o clasă ar trebui să aibă doar o singură intrebuintare.

O clasă ar trebui să fie ușor de extins fără a modifica clasa însăși

Fie q (x) o proprietate demonstrata pentru obiectele de tip x de tipul T. Atunci q (y) ar trebui să fie demonstrata pentru obiectele y de tip S unde S este un subtip a lui T. Toate acestea spun că fiecare subclasă / clasă derivată ar trebui să fie substituibile pentru clasa lor de bază / mamă.

Un client nu ar trebui niciodată obligat să implementeze o interfață pe care nu o folosește sau clienții să nu fie forțați să depindă de metodele pe care nu le utilizează.

Entitățile trebuie să depindă de abstractizări, nu de concretizari. Acest lucru afirmă că modulul de nivel înalt nu trebuie să depindă de modulul de nivel scăzut, dar ele ar trebui să depindă de abstractizări.

Criticile comune ale SOLID:
Principiile SOLID sunt vagi
SOLID duce la un cod complex
SOLID-ul este prea ideal
SOLID este o strategie de marketing

In ciuda criticilor de mai sus setul de principii a dus la o mai buna dezvoltare a aplicatiilor si a proiectelor a caror activitate de dezvoltare se desfasoara in echipe multiple posibil pozitionate in diferite colturi ale lumii. Intradevar aceasta performanta a fost realizata recurgand la anumite compromisuri cum ar fi complexitatea ridicata a scriptului si timpul de finalizare crescut datorita respingerii si reevaluari continue a bucatilor de cod care nu respecta aceste principii adoptate la nivel regula ceea ce uneori este o greseala tinand cont ca SOLID este un set de principii ci nu un set de reguli.

Stefan IOANA