BUDEA Sebastian-Marian
anul IV, IS, grupa 1.2

Principiile S.O.L.I.D

Principiile de proiectare și programare orientată pe obiect s-au dezvoltat în timp, pe baza problemelor practice şi a experienței dobândite de către alți programatori. Aceste principii au fost formulate şi standardizate, în cele din urmă rezultând o serie destul de lungă de principii de design. Dintre acestea, cele mai aplicate şi cunoscute sunt cele cinci care au fost rezumate sub acronimul SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation şi Dependency inversion).

Principiul responsabilității unice (SRP)

> O clasă trebuie să aibă întotdeauna o singură responsabilitate şi numai una

Principiul deschis/închis (OCP)

> Clasele trebuie să fie deschise (open)pentru extensii dar totuși închise (closed) pentru modificări

Principiul de substituție Liskov (LSP)

> Obiectele pot fi înlocuite oricând cu instanțe ale claselor derivate fără ca acest lucru să afecteze funcționalitatea

Principiul de segregare a interfețelor (ISP)

> Mai multe interfețe specializabile sunt oricând de preferat unei singure interfețe generale, prin urmare obiectele nu trebuie obligate sa implementeze metode care nu le sunt utile

Principiul de inversare a dependențelor (DIP)

> O clasă trebuie să depindă de abstractizări, niciodată de obiecte concrete

Arhitectura software

Arhitectura software acoperă întreaga funcţionalitate şi legăturile între componentele sistemelor software de mari dimensiuni. O privire din punct de vedere al arhitecturii, asupra unui sistem, este abstractă, dezvăluind detalii de implementare, algoritmi şi structuri de date, dar concentrându-se pe interacţiunea şi funcţionalitatea componentelor stabilite (privite la şi „black boxes”).

Arhitectura este un design de nivel superior. Această definiţie poate fi adevărată dar nu şi reciprocă: diferite caracteristici ale designului software nu sunt arhitecturale, cum ar fi stabilirea tipurilor de date încapsulate; interfaţa acestor tipuri de date ţine de arhitectură, dar nu şi alegerea efectivă a acestor date. Ea reprezintă structura componentelor unui sistem software, relaţiile dintre ele, principiile şi direcţiile dominante legate de designul şi funcţionalitatea lor. Orice sistem software are o arhitectură care poate fi analizată independent de funcţionalitatea şi procesele pentru care a fost creată. De aceea, „principiile şi direcţiile dominante de designul şi funcţionalitatea lor” se referă la nişte informaţii care pot fi dovedite a fi esenţiale şi un semn de bună practică, dar totuşi, nu se pot considera ca fiind parte integrantă din arhitectură.