PRINCIPIUL SEGREGARII INTERFETELOR - PSI

Clientii nu trebuie fortati sa depinda de interfete pe care nu le folosesc.

Este mai bine sa existe mai multe interfete specializate pe fiecare client decat una de interes general.

Impactul modificarii unei interfete va fi mai mic daca interfata este mai mica.

PRINCIPIUL ECHIVALENTEI REFOLOSIRE/FOLOSIRE - PER

Granularitatea de refolosire este granularitatea de distributie.

Doar componentele ce pot fi usor urmarite pot fi refolosite efficient.

Un element software reutilizabil nu poate fi refolosit cu adevarat daca nu este gestionat de un sistem de distributie.

In practica, nu exista clasa reutilizabila fara garantia notificarii, sigurantei si asistentei.

Trebuie integrat intregul modul.

Fie toate clasele unui pachet sunt reutilizabile, fie niciuna nu este.

PRINCIPIUL REUTILIZARII COMUNE - PRC

Toate clasele unui pachet trebuie refolosite impreuna.

Daca refolosim o clasa din pachet le refolosim pe toate.

Pachetele de component reutilizabile trebuie grupate dupa utilizarea asteptata, nu dupa functionalitate comuna sau alta clasificare arbitrara.

Cand se depinde de un pachet se depinde de fiecare clasa din acesta si in consecinta trebuie sa grupam in pachet clase ce pot fi refolosite impreuna.

PRINCIPIUL INCHIDERII COMUNE - PIC

Clasele unui pachet trebuie sa fie inchise fata de aceleasi tipuri de modificari.

O modificare ce afecteaza pachetul afecteaza toate clasele acelui pachet.

Clasele care se modifica impreuna trebuie grupate la un loc pentru a limita dispersarea modificarilor intre pachete.

Modificarile trebuie sa afecteze un numar minim de pachete.

Clasele unui pachet trebuie sa fie coezive si pentru un anumit tip de notificare fie se modifica toate clasele unei component fie niciuna.

PRINCIPIUL DEPENDENTELOR STABILE - PDS

Dependentele intre component trebuie sa fie in directia stabilitatii.

O componenta trebuie sa depinda de componente mai stabile decat ea.

Stabilitatea este o masura a dificultatii schimbarii unui modul.

Intrucat arhitectura si deciziile de design de nivel inalt nu se modifica frecvent, ar trebui plasate in pachete stabile. Aceste pachete ar trebui sa contina clase ce pot fi extinse, fara sa fie modificate, deci clase abstracte.

PRINCIPIUL ABSTRACTIUNILOR STABILE - PAS

Abstractizarea unui pachet trebuie sa fie proportionlala cu stabilitatea sa.

Pachetele maximal stabile trebuie sa fie maximal abstracte.

Pachetele instabile trebuie sa fie concrete.

Arhitectura ideala ar trebui sa contina pachetele instabile in partea superioara iar pachetele instabile in partea inferioara.

Pachetele stabile trebuie sa fie usor de extins si dificil de modificat.

PRINCIPIUL DESCHIS-INCHIS - PDI

Entitatile software trebuie sa fie deschise la extinderi si inchise la modificari.

Comportamentul modelului poate fi extins.

Codul sursa al modulului nu trebuie modificat.

Inchiderea se obtine prin folosirea abstractizarii.

Toate clasele membre trebuie sa fie private.

Nu se declara variabile globale.

Modificarea datelor publice pot provoca modificari in cascada.

PRINCIPIUL SUBSTITUTIEI LISKOV - PSL

Prin mostenire, orice proprietate adevarata despre obiectele supertipului trebuie sa ramana adevarata despre obiectele subtipului.

Functiile ce folosesc referinte la clasele de baza trebuie sa poata folosi obiecte ale claselor derivate.

Oriunde se face referire la o clasa in cod, orice subclasa a acesteia trebuie sa fie substituibila.

Este illegal ca o clasa derivate sa suprascrie o metoda a clasei de baza printr-o metoda NOP (no operation).

Solutiile sunt: relatia de derivare inversa , extragerea unei clase de baza comune.

PRINCIPIUL INVERSARII DEPENDENTELOR – PID

Modulele de nivel inalt nu trebuie sa depinda de modulele de nivel scazut. Ambele trebuie sa depinda de abstractiuni.

Abstractiunile nu trebuie sa depinda de detalii.

Detaliile trebuie sa depinda de abstractiuni.

O clasa de baza in ierarhie nu trebuie sa-si stie subclasele.

Modulele detaliate trebuie sa depinda de abstractiuni si nimic nu trebuie sa depinda de ele.

Avantajul proiectarii centrate pe interfete este ca tind sa se schimbe mai rar.

Trebuie evitate dependentele tranzititve, adica structurile in care straturi de nivel inalt depind de abstractiuni de nivel scazut.

Dependentele tranzitive pot fi eliminate prin folosirea mostenirii si a claselor stramos abstracte.