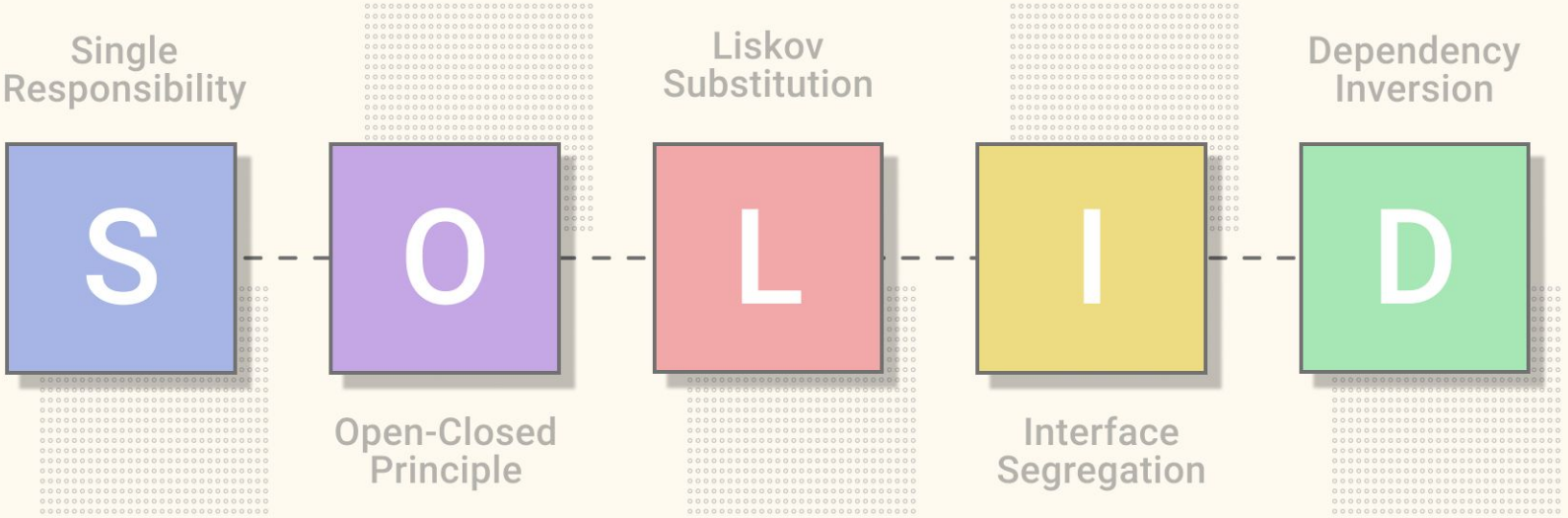
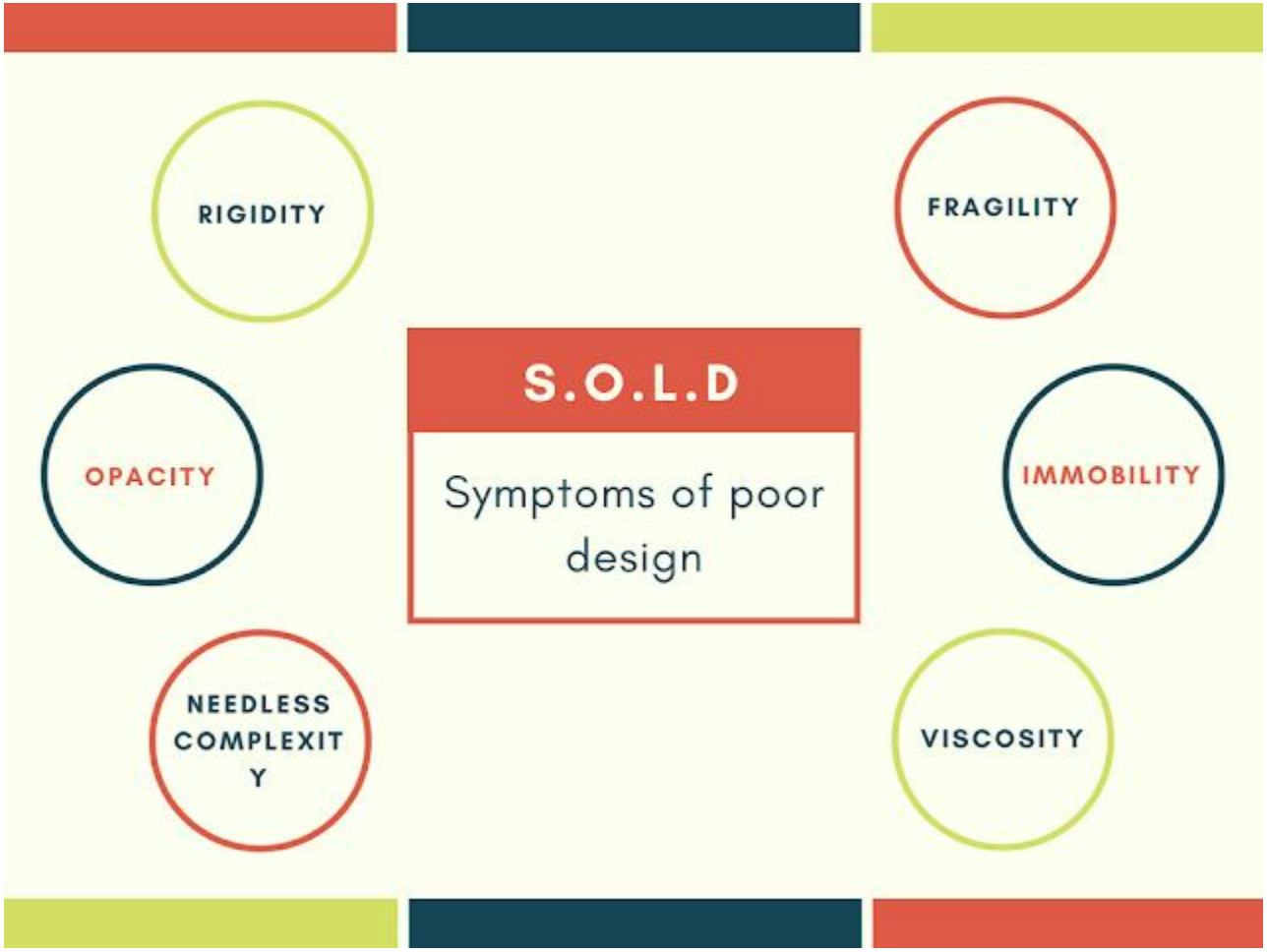
SOLID

Robert C. Martin



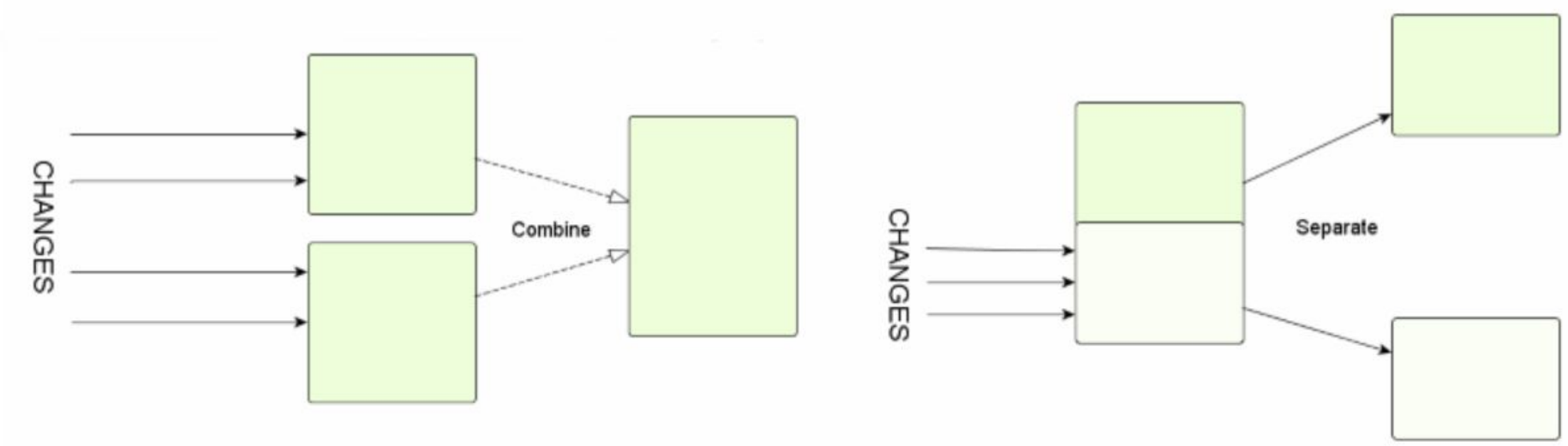
Michael Fathers:


RIGIDITY
FRAGILITY
S.O.L.D
Symptoms of poor design
OPACITY
IMMOBILITY
NEEDLESS COMPLEXITY
VISCOSITY

S - Single responsibility principle (SRP)

A class should have one, and only one, reason to change”

“Princípio da responsabilidade única”

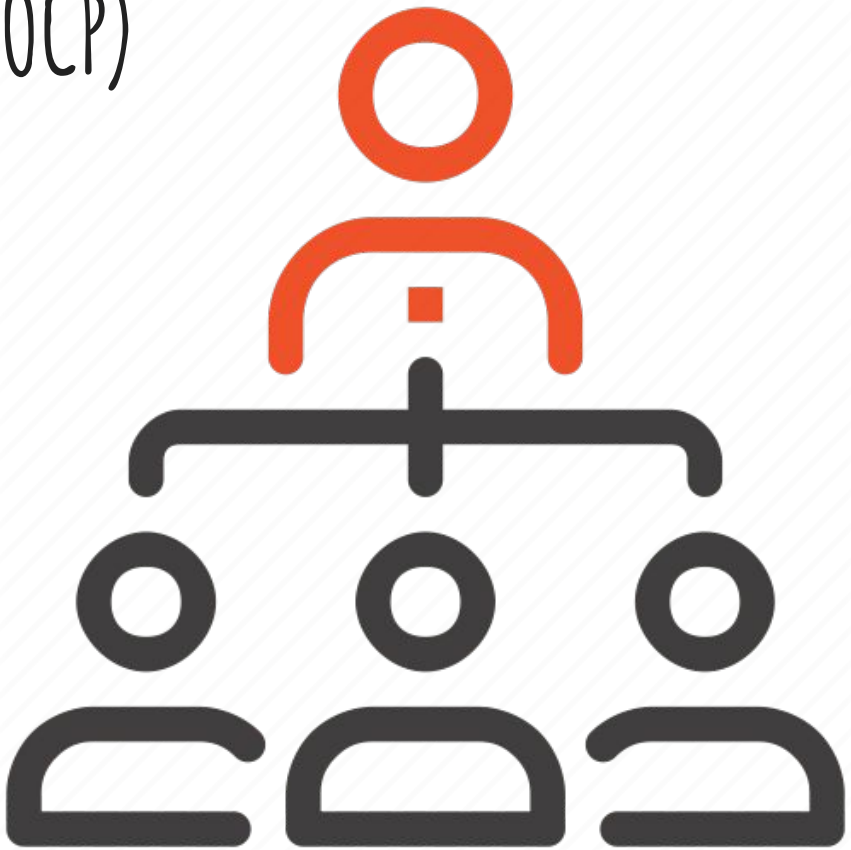


Prática

Is your code SOLID enough?

O - Open closed principle (OCP)

Software entities should be open for extension, but closed for modification



Prática

Is your code SOLID enough?

L - Liskov substitution principle (LSP)

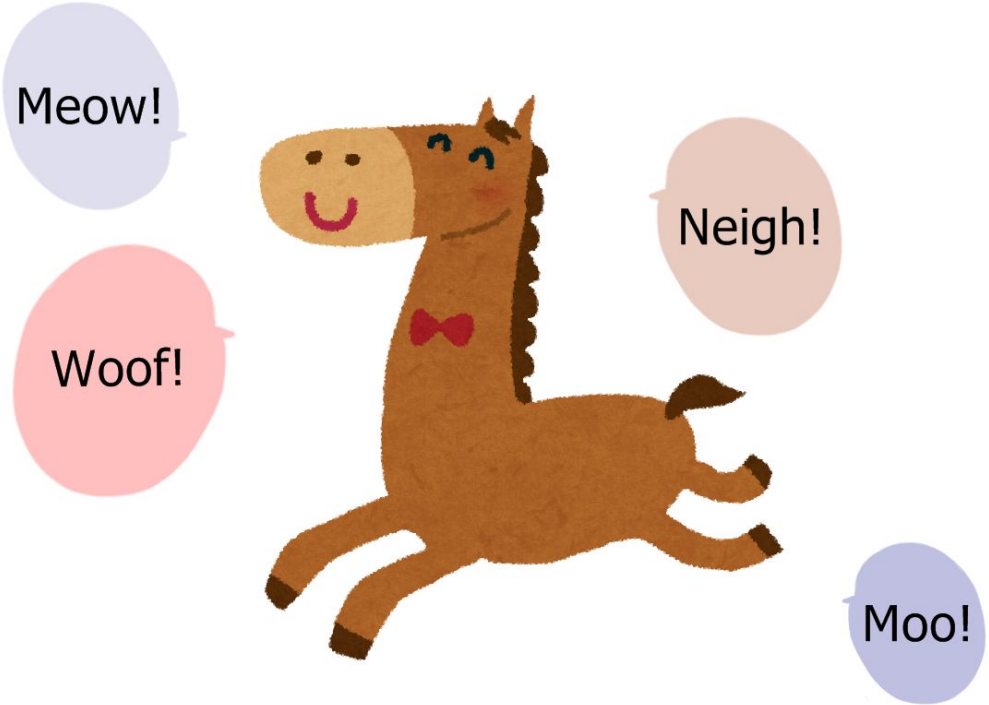
Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program

Prática

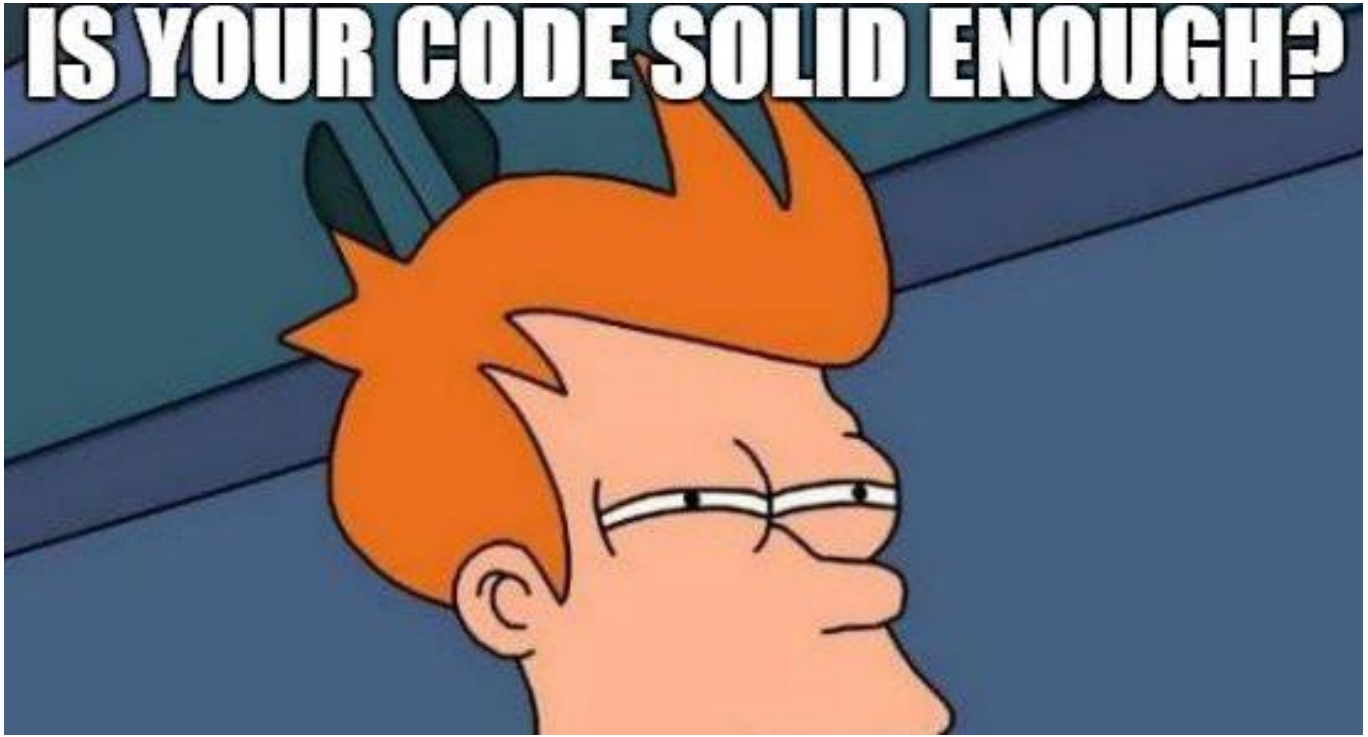
Is your code SOLID enough?

I - Integration segregation Principle - (ISP)

Many client-specific interfaces are better than one general-purpose interface

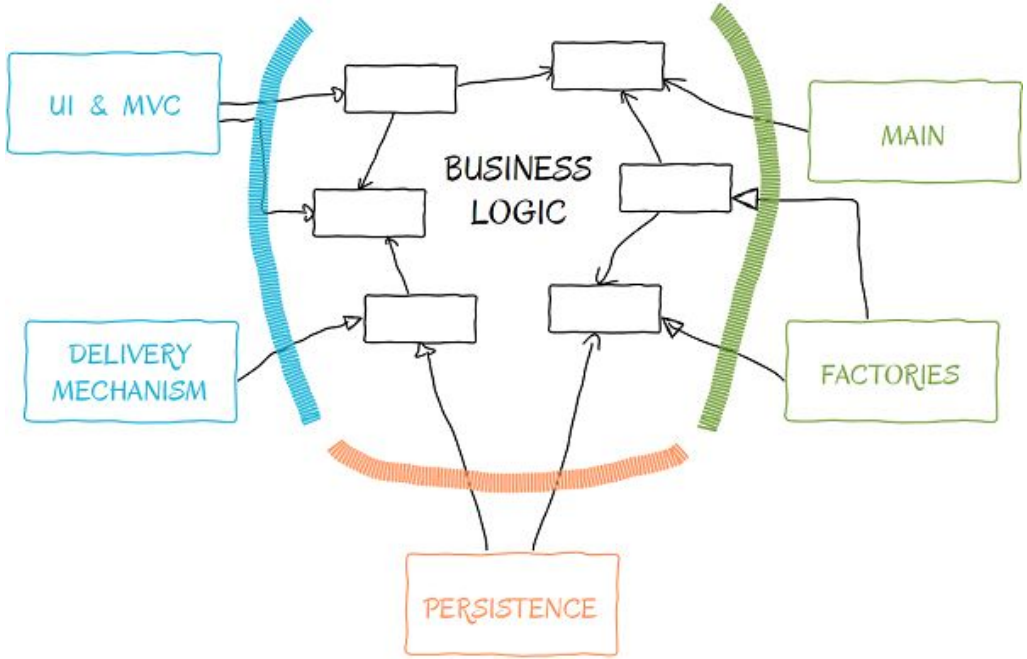


Prática



D - Dependency Inversion Principle (DIP)

- *High-level modules should not depend on low-level modules. Both should depend on abstractions.*
- *Abstractions should not depend on details. Details should depend on abstractions.*



Prática

Is your code SOLID enough?

SOLID

REVISITED

http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

The first five principles are principles of *class design*. They are:

SRP	The Single Responsibility Principle	*A class should have one, and only one, reason to change.*
OCP	The Open Closed Principle	*You should be able to extend a classes behavior, without modifying it.*
LSP	The Liskov Substitution Principle	*Derived classes must be substitutable for their base classes.*
ISP	The Interface Segregation Principle	*Make fine grained interfaces that are client specific.*
DIP	The Dependency Inversion Principle	*Depend on abstractions, not on concretions.*

The next six principles are about packages. In this context a package is a binary deliverable like a .jar file, or a dll as opposed to a namespace like a java package or a C++ namespace.

The first three package principles are about package *cohesion*, they tell us what to put inside packages:

REP	The Release Reuse Equivalency Principle	*The granule of reuse is the granule of release.*
CCP	The Common Closure Principle	*Classes that change together are packaged together.*
CRP	The Common Reuse Principle	*Classes that are used together are packaged together.*

The last three principles are about the couplings between packages, and talk about metrics that evaluate the package structure of a system.

ADP	The Acyclic Dependencies Principle	*The dependency graph of packages must have no cycles.*
SDP	The Stable Dependencies Principle	*Depend in the direction of stability.*
SAP	The Stable Abstractions Principle	*Abstractness increases with stability.*