**Clean Code**

**A Handbook of Agile**

**software Craftsmanship**

"In short, a WHO Creates clean code programmer is an artist WHO can transform a blank canvas into an elegant code system." (Martin, 2009)

**Chapter 8**

Boundaries

The code to third gives us more functionality in less time and it is important to carry out the necessary tests to understand we achieve third-party code, called learning tests. These tests are free and profitable, we provide greater accessibility for handling and testing. Learning these tests indicate that the third package we are using works as expected. If the third party package changes, we'll know at that moment as the tests previously performed showed that the desired function is no longer performed.

It is better control and manage something you already know some of which do not depend.

**Chapter 9**

Unit Tests

Laws to create unit tests or test-driven development (GDP):

1. No need to create production code until you have created a unit test to fail.

2. should not create more than one test unit as failed and not enough to compile a failure is considered.

3. should not create more production code than necessary to overcome the current test failure.

Yel code tests are created jointly, taking into account first and then test the code are analyzed. So several tests that help make our program better and more efficient will be generated. The more tests are done to the code, our program will be of better quality.

Tests should change according to the evolution of the program, so it is important that the tests are updated and clean. The tests help the code is flexible and can be maintained and reused. Without clean tests, stagnation is caused in the code and failures as these tests are not the be changed in relation to the current code.

The most important element is a test code readability.

FIRST:

They are the 5 elements that must follow a clean test:

**Quick**: Tests must be quick because if they are not detected no problems.

**Independence**Must be able to run each test separately and in any order. If you depend on each other, when one fails a row fault is generated.

**Repetition**: Tests can be run in any environment.

**Validation** Automatic: must be of Boolean type, if you pass true or false but.

**Puntuality**: Tests must be created at the time prices before the code.

**Chapter 10**

Classes

They talk about accessibilities can have our attributes and methods in a class, whether private, public or protected. Encapsulation serves to hide certain information. Classes should be small. The name of the class must specify its functionality. The more variables handled more cohesive method will have a class, and that is what is sought. Achieve cohesion shows that each method, variable are interrelated and have a logical order. If any changes are made it is likely to change all the code that's why classes should have an order and thus change the code will be easier. There are concrete classes which are the basis of our code and abstract classes that represent concepts.

**Chapter 11**

Systems

The separation of concerns is one of the most important techniques to be applied and these help us have a cleaner code and follow an order. Code startup process should not be confused with execution. The main function creates the necessary objects for the system passes to another function and this is responsible for performing all running the program. A good programmer must consider the architecture that must have its program because it offers agility. At all levels of abstraction the objects must be clear.

**Chapter 12**

Emergence

A design is simple if it meets these four rules:

Runs all tests: a system that meets all the tests is called testable system. This causes small class designs. If our system can prove we can create better designs. Creating tests allows us to better design our program.

It does not contain duplicate or delete duplicate: it is possible to have several lines that fulfill the same function as each other and are simply worsen our code, it is important to define what each line and verify that there are no similar.

Expresses the intention of the programmer: the code must be clear and understandable for programmers and other programmers, although it is true that when you are programming makes it understandable for himself but bear in mind that code can be useful for another programmer and need to understand what the operation.

Refactor: to the increase in code lines is important to take the time to identify whether our design relates, if you have some aspect inconforme clean it and re-run the tests