OOP HW18 Software Engineering Principles

DATE: 01th february 2021

3 ALVAREZ RAMIREZ MICHELLE ESTEFANIA

THE SOLID PRINCIPLES allow to have a quality software:

- **S**ingle Responsibility Principle.- Each class should be responsible for only one part of the system's functionality.
- **O**pen/Closed Principle.- A class system is open to extension and closed to modification., if Public methods (the abstractions) are declared using interfaces or Implementations and concrete classes inherit the public method declarations from the interfaces,
- **L**iskov Substitution Principle.- if S is a specialization of T, an S object must be able to do everything any T object can do
- **I**nterface Segregation Principle.- An interface is a ”window” or “portal” into the functionality of a component. An interface represents public methods of a component.
- **D**ependency Inversion Principle.- Organize the system into layers. Abstractions should not depend on details. High-level modules should not depend on low-level modules

MULTI-PARADIGM SOFTWARE ENGINEERING

A program must be built in the less time and must be efficiency, security and understandability.

BEST PRACTICES, PATTERNS, AND IDIOMS. They are techniques that help developers adhere to principles, without having to consider the details of a situation at a theoretical level.

You can integrate 3 core principles (rule for creating software with certain desirable characteristics):

Modularity.- It exists in a software system when it is comprised of loosely coupled and cohesive components. Can improve understandability, testability maintainability, reliability, security, extensibility, and reuse.

Abstraction.- The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context

Encapsulation.- The private implementation details of a component must be insulated so they cannot be accessed or modified by other components. Doing so will lead to better testability, maintainability, and reliability. It will also help with a clear separation of concerns and avoid accidental coupling.

Modularity deals with the decomposition of system into components, whereas abstraction and encapsulation deal with individual components. Therefore, modularity cannot be subsumed by either the other two. But, Abstraction and encapsulation might be considered duals of each other.